

DYNAMICALLY DETERMINING APPROPRIATE  
COMPUTER USER INTERFACES

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of U.S. Provisional Application No. 60/240,671 (Attorney Docket Nos. TG1003 and 294438006US00), filed October 16, 2000; of U.S. Provisional Application No. 60/240,682 (Attorney Docket Nos. TG1004 and 294438006US01), filed October 16, 2000; of U.S. Provisional Application No. 60/240,687 (Attorney Docket Nos. TG1005 and 294438006US02), filed October 16, 2000; of U.S. Provisional Application No. 60/240,689 (Attorney Docket Nos. TG1001 and 294438006US03), filed October 16, 2000; of U.S. Provisional Application No. 60/240,694 (Attorney Docket Nos. TG1013 and 294438006US04), filed October 16, 2000; of U.S. Provisional Application No. 60/311,181 (Attorney Docket Nos. 145 and 294438006US06), filed August 9, 2001; of U.S. Provisional Application No. 60/311,148 (Attorney Docket Nos. 146 and 294438006US07), filed August 9, 2001; of U.S. Provisional Application No. 60/311,151 (Attorney Docket Nos. 147 and 294438006US08), filed August 9, 2001; of U.S. Provisional Application No. 60/311,190 (Attorney Docket Nos. 149 and 294438006US09), filed August 9, 2001; of U.S. Provisional Application No. 60/311,236 (Attorney Docket Nos. 150 and 294438006US10), filed August 9, 2001; and of U.S. Provisional Application No. 60/323,032 (Attorney Docket Nos. 135 and 294438006US05), filed September 14, 2001, each of which are hereby incorporated by reference in their entirety.

TECHNICAL FIELD

[0002] The following disclosure relates generally to computer user interfaces, and more particularly to various techniques for dynamically determining an appropriate user interface, such as based on a current context of a user of a wearable computer.

## BACKGROUND

[0003] Current user interfaces (UIs) often use a windows, icons, menus, and pointers (WIMP) interface. While WIMP interfaces have proved useful for some users of stationary desktop computers, a WIMP interface is not typically appropriate for other users (e.g., users that are non-stationary and/or users of other types of computing devices). One reason that WIMP interfaces are inappropriate in other situations is that they make several inappropriate assumptions about the user's situation, including: (a) that the user's computing device has a significant amount of screen real estate available for the UI; (b) that interaction, not digital information, is the user's primary task (e.g., that the user is willing to track a pointer's movement, hunt down a menu item or button, find an icon, and/or immediately receive and respond to information being presented); and (c) that the user can and should explicitly specify how and when to change the interface (e.g., to adapt to changes in the user's environment).

[0004] Moreover, what limited controls are available to the user in a WIMP interface (e.g., manually changing the entire computer display's brightness or audio volume) are typically complicated (e.g., system controls are not integrated in the control mechanisms of the computing system - instead, users must go through multiple layers of system software), inflexible (e.g., user preferences do not apply to different input and output (I/O) devices), non-automated (e.g., UIs do not typically respond to context changes without direct user intervention), not user-extensible (e.g., new devices cannot be integrated into existing preferences), not user-programmable (e.g., users cannot modify underlying logic used), and difficult to share (e.g., there is a lack of integration, which means preference for logic used cannot be conveniently stored and exported to other computers), as well as suffering from various other problems.

[0005] A computing system and/or an executing software application that were able to dynamically modify a UI during execution so as to appropriately reflect current conditions would provide a variety of benefits. However, to perform such dynamic modification of a UI, whether by choosing between existing options and/or by creating a custom UI, such a system and/or software may need to be able to determine and respond to a variety of complex current UI needs. For instance, in a situation in which the user requires that the input to the computing environment be private, the computer-assisted task is complex, and the user has access to a head-mounted display (HMD) and a keyboard, the UI needs are different than a situation in which the user does not require any privacy, has access to a desktop computer with a monitor, and the computer-assisted task is simple.

[0006] Unfortunately, current computing systems and software applications (including WIMP interfaces) do not explicitly model sufficient UI needs (e.g., privacy, safety, available I/O devices, learning style, etc.) to allow an optimal or near-optimal UI to be dynamically determined and used during execution. In fact, most computing systems and software applications do not explicitly model any UI needs, and make no attempt to dynamically modify their UI during execution to reflect current conditions.

[0007] Some current systems do attempt to provide modifiability of UI designs in various limited ways that do not involve modeling such UI needs, but each fail for one reason or another. Some such current techniques include:

- changing UI design based on device type;
- specifying explicit user preferences; and
- changing UI output by selecting a platform at compile-time.

Unfortunately, none of these techniques address the entire problem, as discussed below.

- [0008] Changing the UI based on the type of device (e.g., providing a personal digital assistant (PDA) with a different UI than a desktop computer or a computer in an automobile) typically involves designing completely separate UIs that are not inter-compatible and that do not react to the user's context. Thus, the user gets a different UI on each computing device that they use, and gets the same UI on a particular device regardless of their situation (e.g., whether they are driving a car, working on an airplane engine, or sitting at a desk).
- [0009] Specifying of user preferences (e.g., as allowed by the Microsoft Windows operating system and some application programs) typically allows a UI to be modified, but in ways that are limited to appearance and superficial functionality (e.g., accessibility, pointers, color schemes, etc.), and requires an explicit user intervention (which is typically difficult and time-consuming to specify) every time that the UI is to change.
- [0010] Changing the type of UI output that will be presented (e.g., pop-up menus versus scrolling lists) based on the underlying software platform (e.g., operating system) that will be used to support the presentation is typically a choice that must be made at compile time, and often involves requiring the UI to be limited to a subset of functionality that is available on every platform to be supported. For example, Geoworks' U.S. Patent No. 5,327,529 describes a system that supports the creation of software applications that can change their appearance in limited manners based on different platforms.
- [0011] Thus, while current systems provide limited modifiability of UI designs, such current systems do not dynamically modify a UI during execution so as to appropriately reflect current conditions. The ability to provide such dynamic modification of a UI would provide significant benefits in a wide variety of situations.



## BRIEF DESCRIPTION OF THE DRAWINGS

- [0012] Figure 1 is a data flow diagram illustrating one embodiment of dynamically determining an appropriate or optimal UI.
- [0013] Figure 2 is a block diagram illustrating an embodiment of a computing device with a system for dynamically determining an appropriate UI.
- [0014] Figure 3 illustrates an example relationship between various techniques related to dynamic optimization of computer user interfaces.
- [0015] Figure 4 illustrates an example of an overall mechanism for characterizing a user's context.
- [0016] Figure 5 illustrates an example of automatically generating a task characterization at run time.
- [0017] Figure 6 is a representation of an example of choosing one of multiple arbitrary predetermined UI designs at run time.
- [0018] Figure 7 is a representation of example logic that can be used to choose a UI design at run time.
- [0019] Figure 8 is an example of how to match a UI design characterization with UI requirements via a weighted matching index.
- [0020] Figure 9 is an example of how UI requirements can be weighted so that one characteristic overrides all other characteristics when using a weighted matching index.
- [0021] Figure 10 is an example of how to match a UI design characterization with UI requirements via a weighted matching index.
- [0022] Figure 11 is a block diagram illustrating an embodiment of a computing device capable of executing a system for dynamically determining an appropriate UI.
- [0023] Figure 12 is a diagram illustrating an example of characterizing multiple UI designs.

[0024] Figure 13 is a diagram illustrating another example of characterizing multiple UI designs.

[0025] Figure 14 illustrates an example UI.

#### DETAILED DESCRIPTION

[0026] A software facility is described below that provides various techniques for dynamically determining an appropriate UI to be provided to a user. In some embodiments, the software facility executes on behalf of a wearable computing device in order to dynamically modify a UI being provided to a user of the wearable computing device (also referred to as a wearable personal computer or "WPC") so that the current UI is appropriate for a current context of the user. In order to dynamically determine an appropriate UI, various embodiments characterize various types of UI needs (e.g., based on a current user's situation, a current task being performed, current I/O devices that are available, etc.) in order to determine characteristics of a UI that is currently optimal or appropriate, characterize various existing UI designs or templates in order to identify situations for which they are optimal or appropriate, and then selects and uses one of the existing UIs that is most appropriate based on the current UI needs. In other embodiments, various types of UI needs are characterized and a UI is dynamically generated to reflect those UI needs, such as by combining in an appropriate or optimal manner various UI building block elements that are appropriate or optimal for the UI needs. A UI may in some embodiments be dynamically generated only if an existing available UI is not sufficiently appropriate, and in some embodiments a UI to be used is dynamically generated by modifying an existing available UI.

[0027] For illustrative purposes, some embodiments of the software facility are described below in which current UI needs are determined in particular ways, in which existing UIs are characterized in various ways, and in which appropriate or

optimal UIs are selected or generated in various ways. In addition, some embodiments of the software facility are described below in which described techniques are used to provide an appropriate UI to a user of a wearable computing device based on a current context of the user. However, those skilled in the art will appreciate that the disclosed techniques can be used in a wide variety of other situations and that UI needs and UI characterizations can be determined in a variety of ways.

[0028]

Figure 1 illustrates an example of one embodiment of an architecture for dynamically determining an appropriate UI. In particular, box 109 represents using an appropriate UI for a current context. When changes in the current context render a previous UI inappropriate or non-optimal, a new UI appropriate or optimal UI can be selected or generated, as is shown in boxes 146 and 155 respectively. In order to enable selection of a new UI that is appropriate or optimal, the characteristics of a UI that is currently appropriate or optimal are determined in box 145 and the characteristics of various existing UIs are determined in box 135 (e.g., in a manual and/or automatic manner). In order to enable the determination of the characteristics of a UI that is currently appropriate or optimal, in the illustrated embodiment the UI requirements of the current task are determined in box 149 (e.g., in a manual and/or automatic manner), the UI requirements corresponding to the user are determined in box 150 (e.g., based on the user's current needs), and the UI requirements corresponding to the currently available I/O devices are determined in box 147. The UI requirements corresponding to the user can be determined in various ways, such as in the illustrated embodiment by determining in box 106 the quantity and quality of attention that the user can currently provide to their computing system and/or executing application. If a new appropriate or optimal UI is to generated in box 155, the generation is enabled in the illustrated embodiment by determining the

characteristics of a UI that is currently appropriate or optimal in box 145, determining techniques for constructing a UI design to reflect UI requirements in box 156 (e.g., by combining various specified UI building block elements), and determining how newly available hardware devices can be used as part of the UI. The order and frequency of the illustrated types of processing can be varied in various embodiments, and in other embodiments some of the illustrated types of processing may not be performed and/or additional non-illustrated types of processing may be used.

[0029] Figure 2 illustrates an example computing device 200 suitable for executing an embodiment of the facility, as well as one or more additional computing device 250s with which the computing device 200 may interact. The computing device 200 includes a CPU 205, various I/O devices 210, storage 220, and memory 230. The I/O devices include a display 211, a network connection 212, a computer-readable media drive 213, and other I/O devices 214.

[0030] Various components 241-248 are executing in memory 230 to enable dynamic determination of appropriate or optimal UIs, as are a UI Applier component 249 to apply an appropriate or optimal UI that is dynamically determined. One or more other application programs 235 may also be executing in memory, and the UI Applier may supply, replace or modify the UIs of those application programs. The dynamic determination components include a Task Characterizer 241, a User Characterizer 242, a Computing System Characterizer 243, an Other Accessible Computing Systems Characterizer 244, an Available UI Designs Characterizer 245, an Optimal UI Determiner 246, an Existing UI Selector 247, and a New UI Generator 248. The various components may use and/or generate a variety of information when executing, such as UI building block elements 221, current context information 222, and current characterization information 223.

[0031]

Those skilled in the art will appreciate that computing devices 200 and 250 are merely illustrative and are not intended to limit the scope of the present invention. Computing device 200 may be connected to other devices that are not illustrated, including through one or more networks such as the Internet or via the World Wide Web (WWW), and many in some embodiments be a wearable computer. In other embodiments, the computing devices may comprise other combinations of hardware and software, including computers, network devices, internet appliances, PDAs, wireless phones, pagers, electronic organizers, television-based systems and various other consumer products that include inter-communication capabilities. In addition, the functionality provided by the illustrated components may in some embodiments be combined in fewer components or distributed in additional components. Similarly, in some embodiments the functionality of some of the illustrated components may not be provided and/or other additional functionality may be available.

[0032]

Those skilled in the art will also appreciate that, while various items are illustrated as being stored in memory or on storage while being used, these items or portions of them can be transferred between memory and other storage devices for purposes of memory management and data integrity. Some or all of the components and their data structures may also be stored (e.g., as instructions or structured data) on a computer-readable medium, such as a hard disk, a memory, a network, or a portable article to be read by an appropriate drive. The components and data structures can also be transmitted as generated data signals (e.g., as part of a carrier wave) on a variety of computer-readable transmission mediums, including wireless-based and wired/cable-based mediums. Accordingly, the present invention may be practiced with other computer system configurations.

- [0033] What follows are various examples of techniques for dynamically determining an appropriate UI, such as by characterizing various types of UI needs and/or by characterizing various existing UI designs or templates in order to identify situations for which they are optimal or appropriate.

## MODELING A COMPUTER USER'S COGNITIVE AVAILABILITY

### USER'S MEANING

- [0034] (the significance and/or implication of things, in the user's mind)
- [0035] Task, Purpose, Activity, Destination, Motivation, Desired Privacy
- [0036] When we assign a type, a friendly name, or description to a thing like place, we support the inference of intention.
- [0037] A grocery store is where activity associated with shopping can be accomplished – it is a characterization, an association of activities, in the mind of the user about a specific place.

### USER'S COGNITION

- [0038] Cognitive / Attention Availability
- [0039] "Change in Cognitive Availability ⇔ Change in Mode of interaction" (could differentiate between 'user doesn't have the cycles' and 'user has them, but does not chose to give them to WPC')
- [0040] "State Info/Compartmentalization ⇔ Complexity of UI"
- [0041] Characterize tasks as PC Aware, or not.

### DIVIDED USER ATTENTION

- [0042] This section will deal primarily with Divided Attention.
- [0043] When performing more than one task at a time, the user can engage in three types of tasks:
- [0044] Focus Tasks: requires the users primary attention
- [0045] An example of a Focus Task is looking at a map.

[0046] Routine Tasks: requires attention from the user, but allows multi-tasking in parallel

[0047] An example of a Routine Task is talking on a cell phone, through the headset.

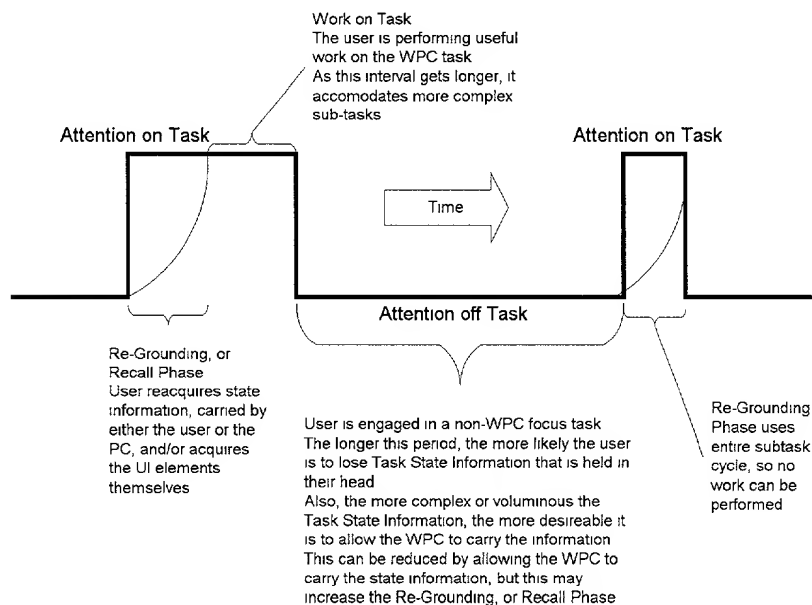
[0048] Awareness Tasks: does not require any significant attention from the user

[0049] For an example of an "Awareness Task", imagine that the rate of data connectivity were represented as the background sound of flowing water. The user would be aware of the rate at some level, without significantly impacting the available User Attention.

[0050] To perform tasks simultaneously, there are three kinds of divided attention; Task Switched, Parallel, and Awareness, as follows:

#### TASK SWITCHING (FOCUS TASK + FOCUS TASK)

[0051] When the user is engaged in more than one Focus Task, the attention is Task Switched. The user performs a compartmentalized subset of one task, interrupts that task, and performs a compartmentalized subset of the other task, as follows:



[0052] Re-Grounding Phase: As the user returns to a Focus Task, they first reacquire any state information associated with the task, and/or acquire the UI elements themselves. Either the user or the WPC can carry the state information.

[0053] Work Phase: Here the user actually performs the sub-task. The longer this phase, the more complex the subtask can be.

[0054] Interruption / Off Task: When the interruption occurs, the user switches from one Focus Task to another task.

[0055] When the duration of Work on Task increases (say, when the user's motion temporarily goes from 30 MPH to 0), then task presentation can more complex. This includes increased context of the steps involved (e.g., view more steps in the Bouncing Ball Wizard) or greater detail of each step (addition of other people's schedule when making appointments).

[0056] The longer the Off Task cycle, the more likely the user is to lose Task State Information that is carried in their head. Also, the more complex or voluminous the Task State Information, the more desirable it becomes to allow the WPC to present the state information. The side effect of using the WPC to present Task State Information is that the Re-Grounding Phase may be lengthened, reducing the Work Phase.

#### PARALLEL

[0057] (Focus Task + Routine) OR (Routine + Routine)

#### BACKGROUND AWARENESS

[0058] The concept of Background Awareness is that a non-focus output stimulus allows the user to monitor information without devoting significant attention or cognition. The stimulus retreats to the subconscious, but the user is consciously aware of an abrupt change in the stimulus.



## COCKTAIL PARTY EFFECT

[0059] In audio, a phenomenon known as the “Cocktail Party Effect” allows a user to listen to multiple background audio channels, as long as the sounds representing each process are distinguishable.

[0060] Experiments have shown that increasing the channels beyond three (3) causes degradation in comprehension. [Stiefelman94]

[0061] Spatial layout (3D Audio) can be used as an aid to audio memory. Focus can be given to a particular audio channel by increasing the gain on that channel.

[0062] Listening and Monitoring have different cognitive burdens.

[0063] The MIT Nomadic Radio Paper “Simultaneous and Spatial Listening” provides additional information on this phenomenon.

## CHARACTERIZING A COMPUTER USER’S UI REQUIREMENTS

[0064] When monitoring and evaluating some or all available characteristics that could cause a UI to change (regardless of the source of the characteristic), it is possible to choose one or more of the most important characteristics upon which to build a UI, and then pass those characteristics to the computing system.

[0065] Considered singularly, many of the characteristics described in this disclosure can be beneficially used to inform a computing system when to change. However, with an extensible system, additional characteristics can be considered (or ignored) at anytime, providing precision to the optimization.

## ATTRIBUTES ANALYZED

[0066] This section describes various modeled real-world and virtual contexts . The described model for optimal UI design characterization includes at least the following categories of attributes when determining the optimal UI design:

[0067] \* All available attributes. The model is dynamic so it can accommodate for any and all attributes that could affect the optimal UI design for a user’s context.

For example, this model could accommodate for temperature, weather conditions, time of day, available I/O devices, preferred volume level, desired level of privacy, and so on.

[0068]       \* Significant attributes. Some attributes have a more significant influence on the optimal UI design than others. Significant attributes include, but are not limited to, the following:

- \* The user can see video.
- \* The user can hear audio.
- \* The computing system can hear the user.
- \* The interaction between the user and the computing system must be private.
- \* The user's hands are occupied.

[0069]       \* Attributes that correspond to a theme. Specific or programmatic. Individual or group.

[0070]       Using even one of these attribute categories can produce a large number of potential UIs. As discussed below, a limited model of user context can generate a large number of distinct situations, each potentially requiring a unique UI design. Despite this large number, this is not a challenge for software implementation. Modern computers can easily handle software implementations of much larger lookup tables.

[0071]       Although this document lists many attributes of a user's tasks and mental and physical environment, these attributes are meant to be illustrative because it is not possible to know all of the attributes that will affect a UI design until run time. The described model is dynamic so it can account for unknown attributes.

[0072]       It is important to note that any of the attributes mentioned in this document are just examples. There are other attributes that can cause a UI to change that

are not listed in this document. However, the dynamic model can account for additional attributes.

## USER CHARACTERIZATIONS

[0073] This section describes the characteristics that are related to the user.

## USER PREFERENCES

[0074] User preferences are a set of attributes that reflect the user's likes and dislikes, such as I/O devices preferences, volume of audio output, amount of haptic pressure, and font size and color for visual display surfaces. User preferences can be classified in the following categories:

[0075] \* Self characterization. Self-characterized user preferences are indications from the user to the computing system about themselves. The self-characterizations can be explicit or implicit. An explicit, self-characterized user preference results in a tangible change in the interaction and presentation of the UI. An example of an explicit, self characterized user preference is "Always use the font size 18" or "The volume is always off." An implicit, self-characterized user preference results in a change in the interaction and presentation of the UI, but it might be not be immediately tangible to the user. A learning style is an implicit self-characterization. The user's learning style could affect the UI design, but the change is not as tangible as an explicit, self-characterized user preference.

[0076] If a user characterizes themselves to a computing system as a "visually impaired, expert computer user," the UI might respond by always using 24-point font and monochrome with any visual display surface. Additionally, tasks would be chunked differently, shortcuts would be available immediately, and other accommodations would be made to tailor the UI to the expert user.

[0077] \* Theme selection. In some situations, it is appropriate for the computing system to change the UI based on a specific theme. For example, a high school student in public school 1420 who is attending a chemistry class could have a UI

appropriate for performing chemistry experiments. Likewise, an airplane mechanic could also have a UI appropriate for repairing airplane engines. While both of these UIs would benefit from hands free, eyes out computing, the UI would be specifically and distinctively characterized for that particular system.

[0078]       \* System characterization. When a computing system somehow infers a user's preferences and uses those preferences to design an optimal UI, the user preferences are considered to be system characterizations. These types of user preferences can be analyzed by the computing system over a specified period on time in which the computing system specifically detects patterns of use, learning style, level of expertise, and so on. Or, the user can play a game with the computing system that is specifically designed to detect these same characteristics.

[0079]       \* Pre-configured. Some characterizations can be common and the UI can have a variety of pre-configured settings that the user can easily indicate to the UI. Pre-configured settings can include system settings and other popular user changes to default settings.

[0080]       \* Remotely controlled. From time to time, it may be appropriate for someone or something other than the user to control the UI that is displayed.

#### EXAMPLE USER PREFERENCE CHARACTERIZATION VALUES

[0081]   This UI characterization scale is enumerated. Some example values include:

- \* Self characterization
- \* Theme selection
- \* System characterization
- \* Pre-configured
- \* Remotely controlled

## THEME

[0082] A theme is a related set of measures of specific context elements, such as ambient temperature, current user task, and latitude, which reflect the context of the user. In other words, theme is a name collection of attributes, attribute values, and logic that relates these things. Typically, themes are associated with user goals, activities, or preferences. The context of the user includes:

- \* The user's mental state, emotional state, and physical or health condition.
- \* The user's setting, situation or physical environment. This includes factors external to the user that can be observed and/or manipulated by the user, such as the state of the user's computing system.
- \* The user's logical and data telecommunications environment (or "cyber-environment," including information such as email addresses, nearby telecommunications access such as cell sites, wireless computer ports, etc.).

[0083] Some examples of different themes include: home, work, school, and so on. Like user preferences, themes can be self characterized, system characterized, inferred, pre-configured, or remotely controlled.

## EXAMPLE THEME CHARACTERIZATION VALUES

[0084] This characteristic is enumerated. The following list contains example enumerated values for theme.

- \* No theme
- \* The user's theme is inferred.
- \* The user's theme is pre-configured.
- \* The user's theme is remotely controlled.
- \* The user's theme is self characterized.
- \* The user's theme is system characterized.

## USER CHARACTERISTICS

[0085] User characteristics include:

- \* Emotional state
- \* Physical state
- \* Cognitive state
- \* Social state

## EXAMPLE USER CHARACTERISTICS CHARACTERIZATION VALUES

[0086] This UI characterization scale is enumerated. The following lists contain some of the enumerated values for each of the user characteristic qualities listed above.

- [0087] \* Emotional state.
- \* Happiness
  - \* Sadness
  - \* Anger
  - \* Frustration
  - \* Confusion
- [0088] \* Physical state
- \* Body
  - \* Biometrics
  - \* Posture
  - \* Motion
  - \* Physical Availability
    - \* Senses
      - \* Eyes
      - \* Ears
      - \* Tactile
      - \* Hands
      - \* Nose
      - \* Tongue
    - \* Workload demands/effects
    - \* Interaction with computer devices
    - \* Interaction with people
    - \* Physical Health
  - \* Environment
    - \* Time/Space
    - \* Objects
    - \* Persons

- \* Audience/Privacy Availability
    - \* Scope of Disclosure
    - \* Hardware affinity for privacy
    - \* Privacy indicator for user
    - \* Privacy indicator for public
    - \* Watching indicator
    - \* Being observed indicator
  - \* Ambient Interference
    - \* Visual
    - \* Audio
    - \* Tactile
- \* Location.
  - \* Place\_name
  - \* Latitude
  - \* Longitude
  - \* Altitude
  - \* Room
  - \* Floor
  - \* Building
  - \* Address
  - \* Street
  - \* City
  - \* County
  - \* State
  - \* Country
  - \* Postal\_Code
- \* Physiology.
  - \* Pulse
  - \* Body\_temperature
  - \* Blood\_pressure
  - \* Respiration
- \* Activity
  - \* Driving
  - \* Eating
  - \* Running
  - \* Sleeping
  - \* Talking
  - \* Typing
  - \* Walking
- \*Cognitive state
  - \* Meaning

- \* Cognition
  - \* Divided User Attention
  - \* Task Switching
  - \* Background Awareness

- \* Solitude
- \* Privacy
  - \* Desired Privacy
  - \* Perceived Privacy
- \* Social Context
- \* Affect

- [0089] \* Social state
- \* Whether the user is alone or if others are present
  - \* Whether the user is being observed (e.g., by a camera)
  - \* The user's perceptions of the people around them and the user's perceptions of the intentions of the people that surround them.
  - \* The user's social role (e.g. they are a prisoner, they are a guard, they are a nurse, they are a teacher, they are a student, etc.)

#### COGNITIVE AVAILABILITY

- [0090] There are three kinds of user tasks: focus, routine, and awareness and three main categories of user attention: background awareness, task switched attention, and parallel. Each type of task is associated with a different category of attention. Focus tasks require the highest amount of user attention and are typically associated with task-switched attention. Routine tasks require a minimal amount of user attention or a user's divided attention and are typically associated with parallel attention. Awareness tasks appeals to a user's precognitive state or attention and are typically associated with background awareness. When there is an abrupt change in the sound, such as changing from a trickle to a waterfall, the user is notified of the change in activity.

#### BACKGROUND AWARENESS

- [0091] Background awareness is a non-focus output stimulus that allows the user to monitor information without devoting significant attention or cognition.



## EXAMPLE BACKGROUND AWARENESS CHARACTERIZATION VALUES

[0092] This characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: the user has no awareness of the computing system/the user has background awareness of the computing system.

[0093] Using these values as scale endpoints, the following list is an example background awareness scale.

- \* No background awareness is available. A user's pre-cognitive state is unavailable.

- \* A user has enough background awareness available to the computing system to receive one type of feedback or status.

- \* A user has enough background awareness available to the computing system to receive more than one type of feedback, status and so on.

- \* A user's background awareness is fully available to the computing system. A user has enough background awareness available for the computing system such that they can perceive more than two types of feedback or status from the computing system.

## EXEMPLARY UI DESIGN IMPLEMENTATIONS FOR BACKGROUND AWARENESS

[0094] The following list contains examples of UI design implementations for how a computing system might respond to a change in background awareness.

- \* If a user does not have any attention for the computing system, that implies that no input or output are needed.

- \* If a user has enough background awareness available to receive one type of feedback, the UI might:

- \* Present a single light in the peripheral vision of a user. For example, this light can represent the amount of battery power available to the

computing system. As the battery life weakens, the light gets dimmer. If the battery is recharging, the light gets stronger.

\* If a user has enough background awareness available to receive more than one type of feedback, the UI might:

\* Present a single light in the peripheral vision of the user that signifies available battery power and the sound of water to represent data connectivity.

\* If a user has full background awareness, then the UI might:

\* Present a single light in the peripheral vision of the user that signifies available battery power, the sound of water that represents data connectivity, and pressure on the skin to represent the amount of memory available to the computing system.

#### TASK SWITCHED ATTENTION

[0095] When the user is engaged in more than one focus task, the user's attention can be considered to be task switched.

#### EXAMPLE TASK SWITCHED ATTENTION

##### CHARACTERIZATION VALUES

[0096] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: the user does not have any attention for a focus task/the user has full attention for a focus task.

[0097] Using these characteristics as the scale endpoints, the following list is an example of a task switched attention scale.

\* A user does not have any attention for a focus task.

\* A user does not have enough attention to complete a simple focus task.

The time between focus tasks is long.

\* A user has enough attention to complete a simple focus task. The time between focus tasks is long.

\* A user does not have enough attention to complete a simple focus task.  
The time between focus tasks is moderately long.

\* A user has enough attention to complete a simple focus task. The time  
between tasks is moderately long.

\* A user does not have enough attention to complete a simple focus task.  
The time between focus tasks is short.

\* A user has enough attention to complete a simple focus task. The time  
between focus tasks is short.

\* A user does not have enough attention to complete a moderately complex  
focus task. The time between focus tasks is long.

\* A user has enough attention to complete a moderately complex focus  
task. The time between focus tasks is long.

\* A user does not have enough attention to complete a moderately complex  
focus task. The time between focus tasks is moderately long.

\* A user has enough attention to complete a moderately complex focus  
task. The time between tasks is moderately long.

\* A user does not have enough attention to complete a moderately complex  
focus task. The time between focus tasks is short.

\* A user has enough attention to complete a moderately complex focus  
task. The time between focus tasks is short.

\* A user does not have enough attention to complete a moderately complex  
focus task. The time between focus tasks is long.

\* A user has enough attention to complete a complex focus task. The time  
between focus tasks is long.

\* A user does not have enough attention to complete a complex focus task.  
The time between focus tasks is moderately long.

\* A user has enough attention to complete a complex focus task. The time between tasks is moderately long.

\* A user does not have enough attention to complete a complex focus task. The time between focus tasks is short.

\* A user has enough attention to complete a complex focus task. The time between focus tasks is short.

\* A user has enough attention to complete a very complex, multi-stage focus task before moving to a different focus task.

### PARALLEL

[0098] Parallel attention can consist of focus tasks interspersed with routine tasks (focus task + routine task) or a series of routine tasks (routing task + routine task).

### EXAMPLE PARALLEL ATTENTION CHARACTERIZATION VALUES

[0099] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: the user does not have enough attention for a parallel task/the user has full attention for a parallel task.

[00100] Using these characteristics as scale endpoints, the following list is an example of a parallel attention scale.

\* A user has enough available attention for one routine task and that task is not with the computing system.

\* A user has enough available attention for one routine task and that task is with the computing system.

\* A user has enough attention to perform two routine tasks and at least of the routine tasks is with the computing system.

\* A user has enough attention to perform a focus task and a routine task. At least one of the tasks is with the computing system.

\* A user has enough attention to perform three or more parallel tasks and at least one of those tasks is in the computing system.

#### PHYSICAL AVAILABILITY

[00101] Physical availability is the degree to which a person is able to perceive and manipulate a device. For example, an airplane mechanic who is repairing an engine does not have hands available to input indications to the computing systems by using a keyboard.

#### LEARNING PROFILE

[00102] A user's learning style is based on their preference for sensory intake of information. That is, most users have a preference for which sense they use to assimilate new information.

#### EXAMPLE LEARNING STYLE CHARACTERIZATION VALUES

[00103] This characterization is enumerated. The following list is an example of learning style characterization values.

- \* Auditory
- \* Visual
- \* Tactile

#### EXEMPLARY UI DESIGN IMPLEMENTATION FOR LEARNING STYLE

[00104] The following list contains examples of UI design implementations for how the computing system might respond to a learning style.

- \* If a user is a auditory learner, the UI might:
  - \* Present content to the user by using audio more frequently.
  - \* Limit the amount of information presented to a user if there is a lot of ambient noise.
- \* If a user is a visual learner, the UI might:
  - \* Present content to the user in a visual format whenever possible.

- \* Use different colors to group different concepts or ideas together.
- \* Use illustrations, graphs, charts, and diagrams to demonstrate content when appropriate.

- \* If a user is a tactile learner, the UI might:
  - \* Present content to the user by using tactile output.
  - \* Increase the affordance of tactile methods of input (e.g. increase the affordance of keyboards).

## SOFTWARE ACCESSIBILITY

[00105] If an application requires a media-specific plug-in, and the user does not have a network connection, then a user might not be able to accomplish a task.

## EXAMPLE SOFTWARE ACCESSIBILITY CHARACTERIZATION VALUES

[00106] This characterization is enumerated. The following list is an example of software accessibility values.

- \* The computing system does not have access to software.
- \* The computing system has access to some of the local software resources.
  - \* The computing system has access to all of the local software resources.
  - \* The computing system has access to all of the local software resources and some of the remote software resources by availing itself to opportunistic user of software resources.
  - \* The computing system has access to all of the local software resources and all remote software resources by availing itself to the opportunistic user of software resources.
  - \* The computing system has access to all software resources that are local and remote.

## PERCEPTION OF SOLITUDE

[00107] Solitude is a user's desire for, and perception of, freedom from input. To meet a user's desire for solitude, the UI can do things like:

- \* Cancel unwanted ambient noise
- \* Block out human made symbols generated by other humans and machines

## EXAMPLE SOLITUDE CHARACTERIZATION VALUES

[00108] This characterization is scalar, with the minimum range being binary. Example binary values, or scalar endpoints, are: no solitude/complete solitude.

[00109] Using these characteristics as scale endpoints, the following list is an example of a solitude scale.

- \* No solitude
- \* Some solitude
- \* Complete solitude

## PRIVACY

[00110] Privacy is the quality or state of being apart from company or observation. It includes a user's trust of audience. For example, if a user doesn't want anyone to know that they are interacting with a computing system (such as a wearable computer), the preferred output device might be a head mounted display (HMD) and the preferred input device might be an eye-tracking device.

## HARDWARE AFFINITY FOR PRIVACY

[00111] Some hardware suits private interactions with a computing system more than others. For example, an HMD is a far more private output device than a desk monitor. Similarly, an earphone is more private than a speaker.

[00112] The UI should choose the correct input and output devices that are appropriate for the desired level of privacy for the user's current context and preferences.

## EXAMPLE PRIVACY CHARACTERIZATION VALUES

[00113] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: not private/private, public/not public, and public/private.

[00114] Using no privacy and fully private as the scale endpoints, the following list is an example privacy characterization scale.

- \* No privacy is needed for input or output interaction
- \* The input must be semi-private. The output does not need to be private.
- \* The input must be fully private. The output does not need to be private.
- \* The input must be fully private. The output must be semi-private.
- \* The input does not need to be private. The output must be fully private.
- \* The input does not need to be private. The output must be semi-private.
- \* The input must be semi-private. The output must be semi-private.
- \* The input and output interaction must be fully private.
- \* Semi-private. The user and at least one other person can have access to or knowledge of the interaction between the user and the computing system.
- \* Fully private. Only the user can have access to or knowledge of the interaction between the user and the computing system.

## EXEMPLARY UI DESIGN IMPLEMENTATION FOR PRIVACY

[00115] The following list contains examples of UI design implementations for how the computing system might respond to a change in task complexity.

- \* If no privacy is needed for input or output interaction:
  - \* The UI is not restricted to any particular I/O device for presentation and interaction. For example, the UI could present content to the user through speakers on a large monitor in a busy office.
- \* If the input must be semi-private and if the output does not need to be private, the UI might:



\* Encourage the user to use coded speech commands or use a keyboard if one is available. There are no restrictions on output presentation.

\* If the input must be fully private and if the output does not need to be private, the UI might:

\* Not allow speech commands. There are no restrictions on output presentation.

\* If the input must be fully private and if the output needs to be semi-private, the UI might:

\* Not allow speech commands (allow only keyboard commands). Not allow an LCD panel and use earphones to provide audio response to the user.

\* If the output must be semi-private and if the input does not need to be private, the UI might:

\* Restrict users to an HMD device and/or an earphone for output. There are no restrictions on input interaction.

\* If the output must be semi-private and if the input does not need to be private, the UI might:

\* Restrict users to HMD devices, earphones, and/or LCD panels. There are no restrictions on input interaction.

\* If the input and output must be semi-private, the UI might:

\* Encourage the user to use coded speech commands and keyboard methods for input. Output may be restricted to HMD devices, earphones or LCD panels.

\* If the input and output interaction must be completely private, the UI might:

\* Not allow speech commands and encourage the user of keyboard methods of input. Output is restricted to HMD devices and/or earphones.

## USER EXPERTISE

- [00116] As the user becomes more familiar with the computing system or the UI, they may be able to navigate through the UI more quickly. Various levels of user expertise can be accommodated. For example, instead of configuring all the settings to make an appointment, users can recite all the appropriate commands in the correct order to make an appointment. Or users might be able to use shortcuts to advance or move back to specific screens in the UI. Additionally, expert users may not need as many prompts as novice users. The UI should adapt to the expertise level of the user.

### EXAMPLE USER EXPERTISE CHARACTERIZATION VALUES

- [00117] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: new user/not new user, not an expert user/expert user, new user/expert user, and novice/expert.

- [00118] Using novice and expert as scale endpoints, the following list is an example user expertise scale.

- \* The user is new to the computing system and to computing in general.
- \* The user is new to the computing system and is an intermediate computer user.
- \* The user is new to the computing system, but is an expert computer user.
- \* The user is an intermediate user in the computing system.
- \* The user is an expert user in the computing system.

### EXEMPLARY UI DESIGN IMPLEMENTATION FOR USER EXPERTISE

- [00119] The following are characteristics of an exemplary audio UI design for novice and expert computer users.

- \* The computing system speaks a prompt to the user and waits for a response.

\* If the user responds in x seconds or less, then the user is an expert. The computing system gives the user prompts only.

\* If the user responds in >x seconds, then the user is a novice and the computing system begins enumerating the choices available.

[00120] This type of UI design works well when more than 1 user accesses the same computing system and the computing system and the users do not know if they are a novice or an expert.

#### LANGUAGE

[00121] User context may include language, as in the language they are currently speaking (e.g. English, German, Japanese, Spanish, etc.).

#### EXAMPLE LANGUAGE CHARACTERIZATION VALUES

[00122] This characteristic is enumerated. Example values include:

- \* American English
- \* British English
- \* German
- \* Spanish
- \* Japanese
- \* Chinese
- \* Vietnamese
- \* Russian
- \* French

#### COMPUTING SYSTEM

[00123] This section describes attributes associated with the computing system that may cause a UI to change.

#### COMPUTING HARDWARE CAPABILITY

[00124] For purposes of user interfaces designs, there are four categories of hardware:

- \* Input/output devices
- \* Storage (e.g. RAM)
- \* Processing capabilities
- \* Power supply

[00125] The hardware discussed in this topic can be the hardware that is always available to the computing system. This type of hardware is usually local to the user. Or the hardware could sometimes be available to the computing system. When a computing system uses resources that are sometimes available to it, this can be called an opportunistic use of resources.

### STORAGE

[00126] Storage capacity refers to how much random access memory (RAM) is available to the computing system at any given moment. This number is not considered to be constant because the computing system might avail itself to the opportunistic use of memory.

[00127] Usually the user does not need to be aware of how much storage is available unless they are engaged in a task that might require more memory than to which they reliably have access. This might happen when the computing system engages in opportunistic use of memory. For example, if an in-motion user is engaged in a task that requires the opportunistic use of memory and that user decides to change location (e.g. they are moving from their vehicle to a utility pole where they must complete other tasks), the UI might warn the user that if they leave the current location, the computing system may not be able to complete the task or the task might not get completed as quickly.

### EXAMPLE STORAGE CHARACTERIZATION VALUES

[00128] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no RAM is available/all RAM is available.

[00129]

Using no RAM is available and all RAM is available, the following table lists an example storage characterization scale.

Scale attribute	Implication
No RAM is available to the computing system	If no RAM is available, there is no UI available.—Or—There is no change to the UI.
Of the RAM available to the computing system, only the opportunistic use of RAM is available.	The UI is restricted to the opportunistic use of RAM.
Of the RAM that is available to the computing system, only the local RAM is accessible	The UI is restricted to using local RAM.
Of the RAM that is available to the computing system, the local RAM is available and the user is about to lose opportunistic use of RAM.	The UI might warn the user that if they lose opportunistic use of memory, the computing system might not be able to complete the task, or the task might not be completed as quickly.
Of the total possible RAM available to the computing system, all of it is available.	If there is enough memory available to the computing system to fully function at a high level, the UI may not need to inform the user. If the user indicates to the computing system that they want a task completed that requires more memory, the UI might suggest that the user change locations to take advantage of additional opportunistic use of memory.

## PROCESSING CAPABILITIES

[00130] Processing capabilities fall into two general categories:

[00131] \* Speed. The processing speed of a computing system is measured in megahertz (MHz). Processing speed can be reflected as the rate of logic calculation and the rate of content delivery. The more processing power a computing system has, the faster it can calculate logic and deliver content to the user.

[00132] \* CPU usage. The degree of CPU usage does not affect the UI explicitly. With current UI design, if the CPU becomes too busy, the UI Typically “freezes” and the user is unable to interact with the computing system. If the CPU usage is too high, the UI will change to accommodate the CPU capabilities. For example, if the processor cannot handle the demands, the UI can simplify to reduce demand on the processor.

### EXAMPLE PROCESSING CAPABILITY

#### CHARACTERIZATION VALUES

[00133] This UI characterization is scalar, with the minimum range being binary. Example binary or scale endpoints are: no processing capability is available/all processing capability is available.

[00134] Using no processing capability is available and all processing capability as scale endpoints, the following table lists an example processing capability scale.

Scale attribute	Implication
No processing power is available to the computing system	There is no change to the UI.
The computing system has access to a slower speed CPU.	The UI might be audio or text only.
The computing system has access to a high speed CPU	The UI might choose to use video in the presentation instead of a

Scale attribute	Implication
	still picture.
The computing system has access to and control of all processing power available to the computing system.	There are no restrictions on the UI based on processing power.

### POWER SUPPLY

[00135] There are two types of power supplies available to computing systems: alternating current (AC) and direct current (DC). Specific scale attributes for AC power supplies are represented by the extremes of the exemplary scale. However, if a user is connected to an AC power supply, it may be useful for the UI to warn an in-motion user when they're leaving an AC power supply. The user might need to switch to a DC power supply if they wish to continue interacting with the computing system while in motion. However, the switch from AC to DC power should be an automatic function of the computing system and not a function of the UI.

[00136] On the other hand, many computing devices, such as wearable personal computers (WPCs), laptops, and PDAs, operate using a battery to enable the user to be mobile. As the battery power wanes, the UI might suggest the elimination of video presentations to extend battery life. Or the UI could display a VU meter that visually demonstrates the available battery power so the user can implement their preferences accordingly.

## EXAMPLE POWER SUPPLY CHARACTERIZATION VALUES

[00137] This task characterization is binary if the power supply is AC and scalar if the power supply is DC. Example binary values are: no power/full power. Example scale endpoints are: no power/all power.

[00138] Using no power and full power as scale endpoints, the following list is an example power supply scale.

- \* There is no power to the computing system.
- \* There is an imminent exhaustion of power to the computing system.
- \* There is an inadequate supply of power to the computing system.
- \* There is a limited, but potentially inadequate supply of power to the computing system.
- \* There is a limited but adequate power supply to the computing system.
- \* There is an unlimited supply of power to the computing system.

## EXEMPLARY UI DESIGN IMPLEMENTATIONS FOR POWER SUPPLY

[00139] The following list contains examples of UI design implementations for how the computing system might respond to a change in the power supply capacity.

- \* If there is minimal power remaining in a battery that is supporting a computing system, the UI might:
  - \* Power down any visual presentation surfaces, such as an LCD.
  - \* Use audio output only.
- \* If there is minimal power remaining in a battery and the UI is already audio-only, the UI might:
  - \* Decrease the audio output volume.
  - \* Decrease the number of speakers that receive the audio output or use earplugs only.



- \* Use mono versus stereo output.

- \* Decrease the number of confirmations to the user.

- \* If there is, for example, six hours of maximum-use battery life available and the computing system determines that the user not have access to a different power source for 8 hours, the UI might:

- \* Decrease the luminosity of any visual display by displaying line drawings instead of 3-dimensional illustrations.

- \* Change the chrominance from color to black and white.

- \* Refresh the visual display less often.

- \* Decrease the number of confirmations to the user.

- \* Use audio output only.

- \* Decrease the audio output volume.

### COMPUTING HARDWARE CHARACTERISTICS

[00140] The following is a list of some of the other hardware characteristics that may be influence what is an optimal UI design.

- \* Cost

- \* Waterproof

- \* Ruggedness

- \* Mobility

[00141] Again, there are other characteristics that could be added to this list. However, it is not possible to list all computing hardware attributes that might influence what is considered to be an optimal UI design until run time.

### BANDWIDTH

[00142] There are different types of bandwidth, for instance:

- \* Network bandwidth

- \* Inter-device bandwidth

## NETWORK BANDWIDTH

[00143] Network bandwidth is the computing system's ability to connect to other computing resources such as servers, computers, printers, and so on. A network can be a local area network (LAN), wide area network (WAN), peer-to-peer, and so on. For example, if the user's preferences are stored at a remote location and the computing system determines that the remote resources will not always be available, the system might cache the user's preferences locally to keep the UI consistent. As the cache may consume some of the available RAM resources, the UI might be restricted to simpler presentations, such as text or audio only.

[00144] If user preferences cannot be cached, then the UI might offer the user choices about what UI design families are available and the user can indicate their design family preference to the computing system.

### EXAMPLE NETWORK BANDWIDTH CHARACTERIZATION VALUES

[00145] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no network access/full network access.

[00146] Using no network access and full network access as scale endpoints, the following table lists an example network bandwidth scale.

Scale attribute	Implication
The computing system does not have a connection to network resources.	The UI is restricted to using local computing resources only. If user preferences are stored remotely, then the UI might not account for user preferences.
The computing system has an unstable connection to network	The UI might warn the user that the connection to remote resources

Scale attribute	Implication
resources.	might be interrupted. The UI might ask the user if they want to cache appropriate information to accommodate for the unstable connection to network resources.
The computing system has a slow connection to network resources.	The UI might simplify, such as offer audio or text only, to accommodate for the slow connection. Or the computing system might cache appropriate data for the UI so the UI can always be optimized without restriction of the slow connection.
The computing system has a high speed, yet limited (by time) access to network resources.	In the present moment, the UI does not have any restrictions based on access to network resources. If the computing system determines that it will lose a network connection, then the UI can warn the user and offer choices, such as does the user want to cache appropriate information, about what to do.
The computing system has a very high-speed connection to network resources.	There are no restrictions to the UI based on access to network resources. The UI can offer text, audio, video, haptic output, and so on.

## INTER-DEVICE BANDWIDTH

[00147] Inter-device bandwidth is the ability of the devices that are local to the user to communicate with each other. Inter-device bandwidth can affect the UI in that if there is low inter-device bandwidth, then the computing system cannot compute logic and deliver content as quickly. Therefore, the UI design might be restricted to a simpler interaction and presentation, such as audio or text only. If bandwidth is optimal, then there are no restrictions on the UI based on bandwidth. For example, the UI might offer text, audio, and 3-D moving graphics if appropriate for the user's context.

### EXAMPLE INTER-DEVICE BANDWIDTH CHARACTERIZATION VALUES

[00148] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no inter-device bandwidth/full inter-device bandwidth.

[00149] Using no inter-device bandwidth and full inter-device bandwidth as scale endpoints, the following table lists an example inter-device bandwidth scale.

Scale attribute	Implication
The computing system does not have inter-device connectivity.	Input and output is restricted to each of the disconnected devices. The UI is restricted to the capability of each device as a stand-alone device.
Some devices have connectivity and others do not.	It depends
The computing system has slow inter-device bandwidth.	The task that the user wants to complete might require more bandwidth that is available among devices. In this case, the UI can offer the user a choice.

Scale attribute	Implication
	Does the user want to continue and encounter slow performance? Or, does the user want to acquire more bandwidth by moving to a different location and taking advantage of opportunistic use of bandwidth?
The computing system has fast inter-device bandwidth.	There are few, if any, restrictions on the interaction and presentation between the user and the computing system. The UI sends a warning message only if there is not enough bandwidth between devices.
The computing system has very high-speed inter-device connectivity.	There are no restrictions on the UI based on inter-device connectivity.

## CONTEXT AVAILABILITY

[00150] Context availability is related to whether the information about the model of the user context is accessible. If the information about the model of the context is intermittent, deemed inaccurate, and so on, then the computing system might not have access to the user's context.

### EXAMPLE CONTEXT AVAILABILITY CHARACTERIZATION VALUES

[00151] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: context not available/context available.

[00152] Using context not available and context available as scale endpoints, the following list is an example context availability scale.

- \* No context is available to the computing system
- \* Some of the user's context is available to the computing system.
- \* A moderate amount of the user's context is available to the computing system.

- \* Most of the user's context is available to the computing system.
- \* All of the user's context is available to the computing system

#### EXEMPLARY UI DESIGN FOR CONTEXT AVAILABILITY

[00153] The following list contains examples of UI design implementations for how the computing system might respond to a change in context availability.

- \* If the information about the model of context is intermittent, deemed inaccurate, or otherwise unavailable, the UI might:
  - \* Stay the same.
  - \* Ask the user if the UI needs to change.
  - \* Infer a UI from a previous pattern if the user's context history is available.
  - \* Change the UI based on all other attributes except for user context (e.g. I/O device availability, privacy, task characteristics, etc.)
  - \* Use a default UI.

#### OPPORTUNISTIC USE OF RESOURCES

[00154] Some UI components, or other enabling UI content, may allow acquisition from outside sources. For example, if a person is using a wearable computer and they sit at a desk that has a monitor on it, the wearable computer might be able to use the desktop monitor as an output device.

#### EXAMPLE OPPORTUNISTIC USE OF RESOURCES

##### CHARACTERIZATION SCALE

[00155] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: no opportunistic use of resources/use of all opportunistic resources.

[00156] Using these characteristics, the following list is an example of an opportunistic use of resources scale.

- \* The circumstances do not allow for the opportunistic use of resources in the computing system.

- \* Of the resources available to the computing system, there is a possibility to make opportunistic use of resources.

- \* Of the resources available to the computing system, the computing system can make opportunistic use of most of the resources..

- \* Of the resources available to the computing system, all are accessible and available.

## CONTENT

[00157] Content is defined as information or data that is part of or provided by a task. Content, in contrast to UI elements, does not serve a specific role in the user/computer dialog. It provides informative or entertaining information to the user. It is not a control. For example a radio has controls (knobs, buttons) used to choose and format (tune a station, adjust the volume & tone) of broadcasted audio content.

[00158] Sometimes content has associated metadata, but it is not necessary.

[00159] Example content characterization values

- \* Quality

- \* Static/streamlined

- \* Passive/interactive

- \* Type

- \* Output device required

- \* Output device affinity

- \* Output device preference

- \* Rendering software

\* Implicit. The computing system can use characteristics that can be inferred from the information itself, such as message characteristics for received messages.

- \* Source. A type or instance of carrier, media, channel or network path

- \* Destination. Address used to reach the user (e.g., a user typically has multiple address, phone numbers, etc.)

- \* Message content. (parseable or described in metadata)

- \* Data format type.

- \* Arrival time.

- \* Size.

- \* Previous messages. Inference based on examination of log of actions on similar messages.

- \* Explicit. Many message formats explicitly include message-characterizing information, which can provide additional filtering criteria.

- \* Title.

- \* Originator identification. (e.g., email author)

- \* Origination date & time

- \* Routing. (e.g., email often shows path through network routers)

- \* Priority

- \* Sensitivity. Security levels and permissions

- \* Encryption type

- \* File format. Might be indicated by file name extension

- \* Language. May include preferred or required font or font type

- \* Other recipients (e.g., email cc field)

- \* Required software



\* Certification. A trusted indication that the offer characteristics are dependable and accurate.

\* Recommendations. Outside agencies can offer opinions on what information may be appropriate to a particular type of user or situation.

## SECURITY

[00160] Controlling security is controlling a user's access to resources and data available in a computing system. For example when a user logs on a network, they must supply a valid user name and password to gain access to resource on the network such as, applications, data, and so on.

[00161] In this sense, security is associated with the capability of a user or outside agencies in relation to a user's data or access to data, and does not specify what mechanisms are employed to assure the security.

[00162] Security mechanisms can also be separately and specifically enumerated with characterizing attributes.

[00163] Permission is related to security. Permission is the security authorization presented to outside people or agencies. This characterization could inform UI creation/selection by giving a distinct indication when the user is presented information that they have given permission to others to access.

## EXAMPLE SECURITY CHARACTERIZATION VALUES

[00164] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints are: no authorized user access/all user access, no authorized user access/public access, and no public access/public access.

[00165] Using no authorized user access and public access as scale endpoints, the following list is an example security scale.

\* No authorized access.

\* Single authorized user access.

\* Authorized access to more than one person

- \* Authorized access for more than one group of people
- \* Public access
- \* Single authorized user only access. The only person who has authorized access to the computing system is a specific user with valid user credentials.
- \* Public access. There are no restrictions on who has access to the computing system. Anyone and everyone can access the computing system.

### EXPOSING CHARACTERIZATION OF USER'S UI NEEDS

[00166] There are many ways to expose user UI need characterizations to the computing system. This section describes some of the ways in which this can be accomplished.

#### NUMERIC KEY

[00167] A context characterization can be exposed to the system with a numeric value corresponding to values of a predefined data structure.

[00168] For instance, a binary number can have each of the bit positions associated with a specific characteristic. The least significant bit may represent the need for a visual display device capable of displaying at least 24 characters of text in an unbroken series. Therefore a UI characterization of decimal 5 would require such a display to optimally display its content.

#### XML TAGS

[00169] A UI's characterization can be exposed to the system with a string of characters conforming to the XML structure.

[00170] For instance, a context characterization might be represented by the following:

```
<Context Characterization>
  <Theme>Work </Theme>
  <Bandwidth>High Speed LAN Network Connection</Bandwidth>
  <Field of View> 28° </Field of View>
```

<Privacy> None </Privacy>

</Context Characterization>

[00171] One significant advantage of the mechanism is that it is easily extensible.

#### PROGRAMMING INTERFACE

[00172] A context characterization can be exposed to the computing system by associating the design with a specific program call.

[00173] For instance:

GetSecureContext can return a handle to the computing system that describes a UI a high security user context.

#### NAME/VALUE PAIRS

[00174] A user's UI needs can be modeled or represented with multiple attributes that each correspond to a specific element of the context (e.g., safety, privacy, or security), and the value of an attribute represents a specific measure of that element. For example, for an attribute that represents the a user's privacy needs, a value of "5 " represents a specific measurement of privacy. Each attribute preferably has the following properties: a name, a value, an uncertainty level, and a timestamp. For example, the name of the privacy attribute may be "User Privacy" and its value at a particular time may be 5. Associated with the current value may be a timestamp of 08/01/2001 13:07 PST that indicates when the value was generated, and an uncertainty level of +/- 1 degrees.

#### HOW TO EXPOSE

##### MANUAL CHARACTERIZATION

[00175] The UI Designer or other person manually and explicitly determines the task characteristic values. For example, XML metadata could be attached to a UI design that explicitly characterizes it as "private" and "very secure."

## MANUAL AND AUTOMATIC CHARACTERIZATION

- [00176] A UI Designer or other person could manually and explicitly determine a task characteristic and the computing system could automatically derive additional values from the manual characterization. For example, if a UI Designer characterized cognitive load as "high," then the computing system might infer that the values of task complexity and task length are "high" and "long," respectively.

## AUTOMATIC CHARACTERIZATION

- [00177] The following list contains some ways in which the previously described methods of task characterization could be automatically exposed to the computing system.
- [00178] \* The computing system examines the structure of the task and automatically evaluates calculates the task characterization method. For example, an application could evaluate how many steps there are in a wizard to task assistant to determine task complexity. The more steps, the higher the task complexity.
- [00179] \* The computing system could apply patterns of use to establish implicit characterizations. For example, characteristics can be based on historic use. A task could have associated with is a list of selected UI designs. A task could therefore have an arbitrary characteristic, such as "activity" with associated values, such as "driving." A pattern recognition engine determines a predictive correlation using a mechanism such as neural networks.

## CHARACTERIZING A TASK'S UI REQUIREMENTS

- [00180] For a system to accurately determine an optimal UI design for a user's current computing context, it should be able to determine the task function including the dialog elements, content, task sequence, user requirements, choices in task and the choices about the task. This disclosure describes an explicit

extensible method to characterize tasks executed with the assistance of a computing system. Computer UIs are designed to allow the interaction between users and computers for a wide range of system configurations and user situations. In general, any task characterizations can be considered if they are exposed in a way that the system can interpret. Therefore there are three aspects :

- [00181]        \* What task characteristics are exposed?
- [00182]        \* What are the methods to characterize the tasks?
- [00183]        \* How are task characteristics exposed to the computing system?

#### TASK CHARACTERIZATIONS

[00184]        A task is a user-perceived objective comprising steps. The topics in this section enumerate some of the important characteristics that can be used to describe tasks. In general, characterizations are needed only if they require a change in the UI design.

[00185]        The topics in this section include examples of task characterizations, example characterization values, and in some cases, example UI designs or design characteristics.

#### TASK LENGTH

[00186]        Whether a task is short or long depends upon how long it takes a target user to complete the task. That is, a short task takes a lesser amount of time to complete than a long task. For example, a short task might be creating an appointment. A long task might be playing a game of chess.

#### EXAMPLE TASK LENGTH CHARACTERIZATION VALUES

[00187]        This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: short/not short, long/not long, or short/long.

[00188] Using short/long as scale endpoints, the list is an example task length scale.

- \* The task is very short and can be completed in 30 seconds or less
- \* The task is moderately short and can be completed in 31-60 seconds.
- \* The task is short and can be completed in 61-90 seconds.
- \* The task is slightly long and can be completed in 91-300 seconds.
- \* The task is moderately long and can be completed in 301-1,200 seconds.
- \* The task is long and can be completed in 1,201-3,600 seconds.
- \* The task is very long and can be completed in 3,601 seconds or more.

#### TASK COMPLEXITY

[00189] Task complexity is measured using the following criteria:

- \* Number of elements in the task. The greater the number of elements, the more likely the task is complex.
- \* Element interrelation. If the elements have a high degree of interrelation, then the more likely the task is complex.
- \* User knowledge of structure. If the structure, or relationships, between the elements in the task is unclear, then the more likely the task is considered to be complex.

[00190] If a task has a large number of highly interrelated elements and the relationship between the elements is not known to the user, then the task is considered to be complex. On the other hand, if there are a few elements in the task and their relationship is easily understood by the user, then the task is considered to be well-structured. Sometimes a well-structured task can also be considered simple.

#### EXAMPLE TASK COMPLEXITY CHARACTERIZATION VALUES

[00191] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: simple/not simple, complex/not

complex, simple/complex, well-structured/not well-structured, or well-structured/complex.

[00192] Using simple/complex as scale endpoints, the list is an example task complexity scale.

- \* There is one, very simple task composed of 1-5 interrelated elements whose relationship is well understood.

- \* There is one simple task composed of 6-10 interrelated elements whose relationship is understood.

- \* There is more than one very simple task and each task is composed of 1-5 elements whose relationship is well understood.

- \* There is one moderately simple task composed of 11-15 interrelated elements whose relationship is 80-90% understood by the user.

- \* There is more than one simple task and each task is composed of 6-10 interrelated whose relationship is understood by the user.

- \* There is one somewhat simple task composed of 16-20 interrelated elements whose relationship is understood by the user.

- \* There is more than one moderately simple task and each task is composed of 11-15 interrelated elements whose relationship is 80-90% understood by the user.

- \* There is one complex task complex task composed of 21-35 interrelated elements whose relationship is 60-79% understood by the user.

- \* There is more than one somewhat complex task and each task is composed of 16-20 interrelated elements whose relationship is understood by the user.

- \* There is one moderately complex task composed of 36-50 elements whose relationship is 80-90% understood by the user.

- \* There is more than one complex task and each task is composed of 21-35 elements whose relationship is 60-79% understood by the user.

- \* There is one very complex task composed of 51 or more elements whose relationship is 40-60% understood by the user.

- \* There is more than one complex task and each task is composed of 36-50 elements whose relationship is 40-60% understood by the user.

- \* There is more than one very complex task and each part is composed of 51 or more elements whose relationship is 20-40% understood by the user.

### EXEMPLARY UI DESIGN IMPLEMENTATION FOR TASK COMPLEXITY

[00193] The following list contains examples of UI design implementations for how the computing system might respond to a change in task complexity.

- \* For a task that is long and simple (well-structured), the UI might:
  - \* Give prominence to information that could be used to complete the task.
  - \* Vary the text-to-speech output to keep the user's interest or attention.
- \* For a task that is short and simple, the UI might:
  - \* Optimize to receive input from the best device. That is, allow only input that is most convenient for the user to use at that particular moment.
  - \* If a visual presentation is used, such as an LCD panel or monitor, prominence may be implemented using visual presentation only.
- \* For a task that is long and complex, the UI might:
  - \* Increase the orientation to information and devices
  - \* Increase affordance to pause in the middle of a task. That is, make it easy for a user to stop in the middle of the task and then return to the task.
- \* For a task that is short and complex, the UI might:
  - \* Default to expert mode.



\* Suppress elements not involved in choices directly related to the current task.

\* Change modality

## TASK FAMILIARITY

[00194] Task familiarity is related to how well acquainted a user is with a particular task. If a user has never completed a specific task, they might benefit from more instruction from the computing environment than a user who completes the same task daily. For example, the first time a car rental associate rents a car to a consumer, the task is very unfamiliar. However, after about a month, the car rental associate is very familiar with renting cars to consumers.

### EXAMPLE TASK FAMILIARITY CHARACTERIZATION VALUES

[00195] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: familiar/not familiar, not unfamiliar/unfamiliar, and unfamiliar/familiar.

[00196] Using unfamiliar and familiar as scale endpoints, the list is an example task familiarity scale.

\* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 1.

\* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 2.

\* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 3.

\* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 4.

\* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 5.

## EXEMPLARY UI DESIGN IMPLEMENTATION FOR TASK FAMILIARITY

[00197] The following list contains examples of UI design implementations for how the computing system might respond to a change in task familiarity.

- \* For a task that is unfamiliar, the UI might:
  - \* Increase task orientation to provide a high level schema for the task.
  - \* Offer detailed help.
  - \* Present the task in a greater number of steps.
  - \* Offer more detailed prompts.
  - \* Provide information in as many modalities as possible.
- \* For a task that is familiar, the UI might:
  - \* Decrease the affordances for help
  - \* Offer summary help
  - \* Offer terse prompts
  - \* Decrease the amount of detail given to the user
  - \* Use auto-prompt and auto-complete (that is, make suggestions based on past choices made by the user).
  - \* The ability to barge ahead is available.
  - \* Use user-preferred modalities.

## TASK SEQUENCE

[00198] A task can have steps that must be performed in a specific order. For example, if a user wants to place a phone call, the user must dial or send a phone number before they are connected to and can talk with another person. On the other hand, a task, such as searching the Internet for a specific topic, can have steps that do not have to be performed in a specific order.

## EXAMPLE TASK SEQUENCE CHARACTERIZATION VALUES

[00199] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: scripted/not scripted, nondeterministic/not nondeterministic, or scripted/nondeterministic.

[00200] Using scripted/nondeterministic as scale endpoints, the following list is an example task sequence scale.

- \* The each step in the task is completely scripted.
- \* The general order of the task is scripted. Some of the intermediary steps can be performed out of order.
- \* The first and last steps of the task are scripted. The remaining steps can be performed in any order.
- \* The steps in the task do not have to be performed in any order.

## EXEMPLARY UI DESIGN IMPLEMENTATION FOR TASK SEQUENCE

[00201] The following list contains examples of UI design implementations for how the computing system might respond to a change in task sequence.

- \* For a task that is scripted, the UI might:
  - \* Present only valid choices.
  - \* Present more information about a choice so a user can understand the choice thoroughly.
  - \* Decrease the prominence or affordance of navigational controls.
- \* For a task that is nondeterministic, the UI might:
  - \* Present a wider range of choices to the user.
  - \* Present information about the choices only upon request by the user.
  - \* Increase the prominence or affordance of navigational controls

## TASK INDEPENDENCE

- [00202] The UI can coach a user through a task or the user can complete the task without any assistance from the UI. For example, if a user is performing a safety check of an aircraft, the UI can coach the user about what questions to ask, what items to inspect, and so on. On the other hand, if the user is creating an appointment or driving home, they might not need input from the computing system about how to successfully achieve their objective.

### EXAMPLE TASK INDEPENDENCE CHARACTERIZATION VALUES

- [00203] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: coached/not coached, not independently executed/independently executed, or coached/independently executed.
- [00204] Using coached/independently executed as scale endpoints, the following list is an example task guidance scale.
- \* The each step in the task is completely scripted.
  - \* The general order of the task is scripted. Some of the intermediary steps can be performed out of order. For example, the first and last steps of the task are scripted and the remaining steps can be performed in any order.
  - \* The steps in the task do not have to be performed in any order.

## TASK CREATIVITY

- [00205] A formulaic task is a task in which the computing system can precisely instruct the user about how to perform the task. A creative task is a task in which the computing system can provide general instructions to the user, but the user uses their knowledge, experience, and/or creativity to complete the task. For example, the computing system can instruct the user about how to write a sonnet.

However, the user must ultimately decide if the combination of words is meaningful or poetic.

#### EXAMPLE TASK CREATIVITY CHARACTERIZATION VALUES

[00206] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints could be defined as formulaic/not formulaic, creative/not creative, or formulaic/creative.

[00207] Using formulaic and creative as scale endpoints, the following list is an example task creativity scale.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 1.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 2.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 3.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 4.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 5.

#### SOFTWARE REQUIREMENTS

[00208] Tasks can be intimately related to software requirements. For example, a user cannot create a complicated database without software.

#### EXAMPLE SOFTWARE REQUIREMENTS

##### CHARACTERIZATION VALUES

[00209] This task characterization is enumerated. Example values include:

- \* JPEG viewer
- \* PDF reader
- \* Microsoft Word

- \* Microsoft Access
- \* Microsoft Office
- \* Lotus Notes
- \* Windows NT 4.0
- \* Mac OS 10

#### TASK PRIVACY

[00210] Task privacy is related to the quality or state of being apart from company or observation. Some tasks have a higher level of desired privacy than others. For example, calling a physician to receive medical test results has a higher level of privacy than making an appointment for a meeting with a co-worker.

#### EXAMPLE TASK PRIVACY CHARACTERIZATION VALUES

[00211] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: private/not private, public/not public, or private/public.

[00212] Using private/public as scale endpoints, the following table is an example task privacy scale.

- \* The task is not public. Anyone can have knowledge of the task.
- \* The task is semi-private. The user and at least one other person have knowledge of the task.
- \* The task is fully private. Only the user can have knowledge of the task.

#### HARDWARE REQUIREMENTS

[00213] A task can have different hardware requirements. For example, talking on the phone requires audio input and output while entering information into a database has an affinity for a visual display surface and a keyboard.

#### EXAMPLE HARDWARE REQUIREMENTS

#### CHARACTERIZATION VALUES

- \* 10 MB available of storage

\* 1 hour of power supply

\* A free USB connection

#### TASK COLLABORATION

[00214] A task can be associated with a single user or more than one user. Most current computer-assisted tasks are designed as single-user tasks. Examples of collaborative computer-assisted tasks include participating in a multi-player video game or making a phone call.

#### EXAMPLE TASK COLLABORATION CHARACTERIZATION VALUES

[00215] This task characterization is binary. Example binary values are single user/collaboration.

#### TASK RELATION

[00216] A task can be associated with other tasks, people, applications, and so on. Or a task can stand alone on it's own.

#### EXAMPLE TASK RELATION CHARACTERIZATION VALUES

[00217] This task characterization is binary. Example binary values are unrelated task/related task.

#### TASK COMPLETION

[00218] There are some tasks that must be completed once they are started and others that do not have to be completed. For example, if a user is scuba diving and is using a computing system while completing the task of decompressing, it is essential that the task complete once it is started. To ensure the physical safety of the user, the software must maintain continuous monitoring of the user's elapsed time, water pressure, and air supply pressure/quantity. The computing system instructs the user about when and how to safely decompress. If this task is stopped for any reason, the physical safety of the user could be compromised.

## EXAMPLE TASK COMPLETION CHARACTERIZATION VALUES

[00219]

Example values are:

- \* Must be completed
- \* Does not have to be completed
- \* Can be paused
- \* Not known

## TASK PRIORITY

[00220]

Task priority is concerned with order. The order may refer to the order in which the steps in the task must be completed or order may refer to the order in which a series of tasks must be performed. This task characteristic is scalar. Tasks can be characterized with a priority scheme, such as (beginning at low priority) entertainment, convenience, economic/personal commitment, personal safety, personal safety and the safety of others. Task priority can be defined as giving one task preferential treatment over another. Task priority is relative to the user. For example, "all calls from mom" may be a high priority for one user, but not another user.

## EXAMPLE TASK PRIVACY CHARACTERIZATION VALUES

[00221]

This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are no priority/high priority.

[00222]

Using no priority and high priority as scale endpoints, the following list is an example task priority scale.

- \* The current task is not a priority. This task can be completed at any time.
- \* The current task is a low priority. This task can wait to be completed until the highest priority, high priority, and moderately high priority tasks are completed.
- \* The current task is moderately high priority. This task can wait to be completed until the highest priority and high priority tasks are addressed.



\* The current task is high priority. This task must be completed immediately after the highest priority task is addressed.

\* The current task is of the highest priority to the user. This task must be completed first.

### TASK IMPORTANCE

[00223] Task importance is the relative worth of a task to the user, other tasks, applications, and so on. Task importance is intrinsically associated with consequences. For example, a task has higher importance if very good or very bad consequences arise if the task is not addressed. If few consequences are associated with the task, then the task is of lower importance.

### EXAMPLE TASK IMPORTANCE CHARACTERIZATION VALUES

[00224] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are not important/very important.

[00225] Using not important and very important as scale endpoints, the following list is an example task importance scale.

\* The task is not important to the user. This task has an importance rating of "1."

\* The task is of slight importance to the user. This task has an importance rating of "2."

\* The task is of moderate importance to the user. This task has an importance rating of "3."

\* The task is of high importance to the user. This task has an importance rating of "4."

\* The task is of the highest importance to the user. This task has an importance rating of "5."

## TASK URGENCY

[00226] Task urgency is related to how immediately a task should be addressed or completed. In other words, the task is time dependent. The sooner the task should be completed, the more urgent it is.

### EXAMPLE TASK URGENCY CHARACTERIZATION VALUES

[00227] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are not urgent/very urgency.

[00228] Using not urgent and very urgent as scale endpoints, the following list is an example task urgency scale.

- \* A task is not urgent. The urgency rating for this task is "1."
- \* A task is slightly urgent. The urgency rating for this task is "2."
- \* A task is moderately urgent. The urgency rating for this task is "3."
- \* A task is urgent. The urgency rating for this task is "4."
- \* A task is of the highest urgency and requires the user's immediate attention. The urgency rating for this task is "5."

### EXEMPLARY UI DESIGN IMPLEMENTATION FOR TASK URGENCY

[00229] The following list contains examples of UI design implementations for how the computing system might respond to a change in task urgency.

- \* If the task is not very urgent (e.g. a task urgency rating of 1, using the scale from the previous list), the UI might not indicate task urgency.

- \* If the task is slightly urgent (e.g. a task urgency rating of 2, using the scale from the previous list), and if the user is using a head mounted display (HMD), the UI might blink a small light in the peripheral vision of the user.

- \* If the task is moderately urgent (e.g. a task urgency rating of 3, using the scale from the previous list), and if the user is using an HMD, the UI might make the light that is blinking in the peripheral vision of the user blink at a faster rate.

\* If the task is urgent, (e.g. a task urgency rating of 4, using the scale from the previous list), and if the user is wearing an HMD, two small lights might blink at a very fast rate in the peripheral vision of the user.

\* If the task is very urgent, (e.g. a task urgency rating of 5, using the scale from the previous list), and if the user is wearing an HMD, three small lights might blink at a very fast rate in the peripheral vision of the user. In addition, a notification is sent to the user's direct line of sight that warns the user about the urgency of the task. An audio notification is also presented to the user.

### TASK CONCURRENCY

[00230] Mutually exclusive tasks are tasks that cannot be completed at the same time while concurrent tasks can be completed at the same time. For example, a user cannot interactively create a spreadsheet and a word processing document at the same time. These two tasks are mutually exclusive. However, a user can talk on the phone and create a spreadsheet at the same time.

### EXAMPLE TASK CONCURRENCY CHARACTERIZATION VALUES

[00231] This task characterization is binary. Example binary values are mutually exclusive and concurrent.

### TASK CONTINUITY

[00232] Some tasks can have their continuity or uniformity broken without comprising the integrity of the task, while other cannot be interrupted without compromising the outcome of the task. The degree to which a task is associated with saving or preserving human life is often associated with the degree to which it can be interrupted. For example, if a physician is performing heart surgery, their task of performing heart surgery is less interruptible than the task of making an appointment.

## EXAMPLE TASK CONTINUITY CHARACTERIZATION VALUES

[00233] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are interruptible/not interruptible or abort/pause.

[00234] Using interruptible/not interruptible as scale endpoints, the following list is an example task continuity scale.

- \* The task cannot be interrupted.
- \* The task can be interrupted for 5 seconds at a time or less.
- \* The task can be interrupted for 6-15 seconds at a time.
- \* The task can be interrupted for 16-30 seconds at a time.
- \* The task can be interrupted for 31-60 seconds at a time.
- \* The task can be interrupted for 61-90 seconds at a time.
- \* The task can be interrupted for 91-300 seconds at a time.
- \* The task can be interrupted for 301-1,200 seconds at a time.
- \* The task can be interrupted 1,201-3,600 seconds at a time.
- \* The task can be interrupted for 3,601 seconds or more at a time.
- \* The task can be interrupted for any length of time and for any frequency.

## COGNITIVE LOAD

[00235] Cognitive load is the degree to which working memory is engaged in processing information. The more working memory is used, the higher the cognitive load. Cognitive load encompasses the following two facets: cognitive demand and cognitive availability.

[00236] Cognitive demand is the number of elements that a user processes simultaneously. To measure the user's cognitive load, the system can combine the following three metrics: number of elements, element interaction, and structure. Cognitive demand is increased by the number of elements intrinsic to the task. The higher the number of elements, the more likely the task is

cognitively demanding. Second, cognitive demand is measured by the level of interrelation between the elements in the task. The higher the interrelation between the elements, the more likely the task is cognitively demanding. Finally, cognitive load is measured by how well revealed the relationship between the elements is. If the structure of the elements is known to the user or if it's easily understood, then the cognitive demand of the task is reduced.

[00237] Cognitive availability is how much attention the user uses during the computer-assisted task. Cognitive availability is composed of the following:

- \* Expertise. This includes schema and whether or not it is in long term memory

- \* The ability to extend short term memory.

- \* Distraction. A non-task cognitive demand.

#### HOW COGNITIVE LOAD RELATES TO OTHER ATTRIBUTES

[00238] Cognitive load relates to at least the following attributes:

- \* Learner expertise (novice/expert). Compared to novices, experts have an extensive schemata of a particular set of elements and have automaticity, the ability to automatically understand a class of elements while devoting little to no cognition to the classification. For example, a novice reader must examine every letter of the word that they're trying to read. On the other hand, an expert reader has built a schema so that elements can be "chunked" into groups and accessed as a group rather than a single element. That is, an expert reader can consume paragraphs of text at a time instead of examining each letter.

- \* Task familiarity (unfamiliar/familiar). When a novice and an expert come across an unfamiliar task, each will handle it differently. An expert is likely to complete the task either more quickly or successfully because they access schemas that they already have and use those to solve the problem/understand

the information. A novice may spend a lot of time developing a new schema to understand the information/solve the problem.

\* Task complexity (simple/complex or well-structured/complex). A complex task is a task whose structure is not well-known. There are many elements in the task and the elements are highly interrelated. The opposite of a complex task is well-structured. An expert is well-equipped to deal with complex problems because they have developed habits and structures that can help them decompose and solve the problem.

\* Task length (short/long). This relates to how much a user has to retain in working memory.

\* Task creativity. (formulaic/creative) How well known is the structure of the interrelation between the elements?

#### EXAMPLE COGNITIVE DEMAND CHARACTERIZATION VALUES

[00239] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are cognitively undemanding/cognitively demanding.

#### EXEMPLARY UI DESIGN IMPLEMENTATION FOR COGNITIVE LOAD

[00240] A UI design for cognitive load is influenced by a tasks intrinsic and extrinsic cognitive load. Intrinsic cognitive load is the innate complexity of the task and extrinsic cognitive load is how the information is presented. If the information is presented well (e.g. the schema of the interrelation between the elements is revealed), it reduces the overall cognitive load.

[00241] The following list contains examples of UI design implementations for how the computing system might respond to a change cognitive load.

\* Present information to the user by using more than one channel. For example, present choices visually to the user, but use audio for prompts.

\* Use a visual presentation to reveal the relationships between the elements. For example if a family tree is revealed, use colors and shapes to represent male and female members of the tree or shapes and colors can be used to represent different family units.

\* Reduce the redundancy. For example, if the structure of the elements is revealed visually, do not use audio to explain the same structure to the user.

\* Keep complementary or associated information together. For example, if creating a dialog box so a user can print, create a button that has the word "Print" on it instead of a dialog box that has a question "Do you want to print?" with a button with the work "OK" on it.

#### TASK ALTERABILITY

[00242] Some task can be altered after they are completed while others cannot be changed. For example, if a user moves a file to the Recycle Bin, they can later retrieve the file. Thus, the task of moving the file to the Recycle Bin is alterable. However, if the user deletes the file from the Recycle Bin, they cannot retrieve it at a later time. In this situation, the task is irrevocable.

#### EXAMPLE TASK ALTERABILITY CHARACTERIZATION VALUES

[00243] This task characterization is binary, with the minimum range being binary. Example binary values or scale endpoints are alterable/not alterable, irrevocable/revocable, or alterable/irrevocable.

#### TASK CONTENT TYPE

[00244] This task characteristic describes the type of content to be used with the task. For example, text, audio, video, still pictures, and so on.

## EXAMPLE CONTENT TYPE CHARACTERISTICS VALUES

[00245] This task characterization is an enumeration. Some example values are:

- \* .asp
- \* .jpeg
- \* .avi
- \* .jpg
- \* .bmp
- \* .jsp
- \* .gif
- \* .php
- \* .htm
- \* .txt
- \* .html
- \* .wav
- \* .doc
- \* .xls
- \* .mdb
- \* .vbs
- \* .mpg

[00246] Again, this list is meant to be illustrative, not exhaustive.

## TASK TYPE

[00247] A task can be performed in many types of situations. For example, a task that is performed in an augmented reality setting might be presented differently to the user than the same task that is executed in a supplemental setting.

## EXAMPLE TASK TYPE CHARACTERISTICS VALUES

[00248] This task characterization is an enumeration. Example values can include:

- \* Supplemental



\* Augmentative

\* Mediated

## METHODS OF TASK CHARACTERIZATION

[00249] There are many ways to expose task characterizations to the system. This section describes some of the ways in which this can be accomplished.

### NUMERIC KEY

[00250] Task characterization can be exposed to the system with a numeric value corresponding to values of a predefined data structure.

[00251] For instance, a binary number can have each of the bit positions associated with a specific characteristic. The least significant bit may represent task hardware requirements. Therefore a task characterization of decimal 5 would indicate that minimal processing power is required to complete the task.

### XML TAGS

[00252] Task characterization can be exposed to the system with a string of characters conforming to the XML structure.

[00253] For instance, a simple and important task could be represented as:  
<Task Characterization> <Task Complexity= "0" Task Length= "9"> </Task Characterization>

[00254] One significant advantage of this mechanism is that it is easily extensible.

### PROGRAMMING INTERFACE

[00255] A task characterization can be exposed to the system by associating a task characteristic with a specific program call.

[00256] For instance:

GetUrgentTask can return a handle to that communicates that task urgency to the UI.

## NAME/VALUE PAIRS

- [00257] A task is modeled or represented with multiple attributes that each correspond to a specific element of the task (e.g., complexity, cognitive load or task length), and the value of an attribute represents a specific measure of that element. For example, for an attribute that represents the task complexity, a value of "5" represents a specific measurement of complexity. Each attribute preferably has the following properties: a name, a value, an uncertainty level, and a timestamp. For example, the name of the complexity attribute may be "task complexity" and its value at a particular time may be 5. Associated with the current value may be a timestamp of 08/01/2001 13:07 PST that indicates when the value was generated, and an uncertainty level of +/- 1 degrees.

## HOW TO EXPOSE TO THE COMPUTING SYSTEM

### MANUAL CHARACTERIZATION

- [00258] The UI Designer or other person manually and explicitly determines the task characteristic values. For example, XML metadata could be attached to a UI design that explicitly characterizes it as "private" and "very secure."

### MANUAL AND AUTOMATIC CHARACTERIZATION

- [00259] A UI Designer or other person could manually and explicitly determine a task characteristic and the computing system could automatically derive additional values from the manual characterization. For example, if a UI Designer characterized cognitive load as "high," then the computing system might infer that the values of task complexity and task length are "high" and "long," respectively.

- [00260] Another manual and automatic characterization is to group together tasks can as a series of interconnected subtasks, creating both a micro-level view of intermediary steps as well as a macro-level view of the method for accomplishing an overall user task. This applies to tasks that range from simple single steps to

complicated parallel and serial tasks that can also include calculations, logic, and nondeterministic subtask paths through the overall task completion process.

[00261] Macro-level task characterizations can then be assessed at design time, such as task length, number of steps, depth of task flow hierarchy, number of potential options, complexity of logic, amount of user inputs required, and serial vs. parallel vs. nondeterministic subtask paths.

[00262] Micro-level task characterizations can also be determined to include subtask content and expected task performance based on prior historical databases of task performance relative to user, task type, user and computing system context, and relevant task completion requirements.

[00263] Examples of methods include:

- \* Add together and utilize a weighting algorithm across the number of exit options from the current state of the procedure.

- \* Calculate depth and size of associated text (more text implying longer time needs and more complexity, and vice versa), graphics, and content types (audio, visual, and other input/output modalities).

- \* Determine number / type of steps and number / type of follow-on calculations affected.

- \* Use associated metadata based on historical databases of relevant actual time, complexity, and user context metrics.

- \* Bound the overall task sequence and associate them as a subroutine, and then all intermediary steps can be individually assessed and added together for cumulative and synergistic characterization of the task. Cumulative characterization will add together specific metrics over all subtasks within the overall task, and synergistic characterization will include user response variables to certain subtask sequences (example: multiple long text descriptions may generally be skimmed by the user to decrease overall time commitment to the

task, thereby providing a sliding scale weight relating text length to actual time to read and understand).

- \* Determine level of input(s) needed by whether the subtask options are predetermined or require independent thought, creation, and input into the system for nondeterministic potential task flow inputs and outcomes.

- \* Pre-set task feasibility factors at design time to include the needs and relative weighting factors for related software, hardware, I/O device availability, task length, task privacy, and other characteristics for task completion and/or for expediting completion of task. Compare these values to real time/run time values to determine expected effects for different value ranges for task characterizations.

#### AUTOMATIC CHARACTERIZATION

[00264] The following list contains some ways in which the previously described methods of task characterization could be automatically exposed to the computing system.

- \* The computing system examines the structure of the task and automatically evaluates calculates the task characterization method. For example, an application could evaluate how many steps there are in a wizard to task assistant to determine task complexity. The more steps, the higher the task complexity.

- \* The computing system could apply patterns of use to establish implicit characterizations. For example, characteristics can be based on historic use. A task could have associated with is a list of selected UI designs. A task could therefore have an arbitrary characteristics, such as "activity" with associated values, such as "driving." A pattern recognition engine determines a predictive correlation using a mechanism such as neural networks.

## CHARACTERIZING I/O DEVICES' UI REQUIREMENTS

### CHARACTERIZED I/O DEVICE ATTRIBUTES

[00265] The described model for optimal UI design characterization includes at least the following categories of attributes when determining the optimal UI design:

[00266] All available attributes. The model is dynamic so it can accommodate for any and all attributes that could affect the optimal UI design for a user's context. For example, this model could accommodate for temperature, weather conditions, time of day, available I/O devices, preferred volume level, desired level of privacy, and so on.

[00267] Significant attributes. Some attributes have a more significant influence on the optimal UI design than others. Significant attributes include, but are not limited to, the following:

- o The user can see video.
- o The user can hear audio.
- o The computing system can hear the user.
- o The interaction between the user and the computing system must be private.
- o The user's hands are occupied.

[00268] Attributes that correspond to a theme. Specific or programmatic. Individual or group.

[00269] The attributes described in this section are example important attributes for determining an optimal UI. Any of the listed attributes can have additional supplemental characterizations. For clarity, each attribute described in this topic is presented with a scale and some include design examples. It is important to note that any of the attributes mentioned in this document are just examples. There are other attributes that can cause a UI to change that are not listed in this

document. However, the dynamic model can account for additional attribute triggers.

#### PHYSICAL AVAILABILITY

- [00270] Physical availability is the degree to which a person is able to perceive and manipulate a device. For example, an airplane mechanic who is repairing an engine does not have hands available to input indications to the computing systems by using a keyboard.

#### I/O DEVICE SELECTION

- [00271] Users may have access to multiple input and output (I/O) devices. Which input or output devices they use depends on their context. The UI should pick the ideal input and output devices so the user can interact effectively and efficiently with the computer or computing device.

#### REDUNDANT CONTROLS

#### PRIVACY

- [00272] Privacy is the quality or state of being apart from company or observation. It includes a user's trust of audience. For example, if a user doesn't want anyone to know that they are interacting with a computing system (such as a wearable computer), the preferred output device might be an HMD and the preferred input device might be an eye-tracking device.

#### HARDWARE AFFINITY FOR PRIVACY

- [00273] Some hardware suits private interactions with a computing system more than others. For example, an HMD is a far more private output device than a desk monitor. Similarly, an earphone is more private than a speaker.
- [00274] The UI should choose the correct input and output devices that are appropriate for the desired level of privacy for the user's current context and preferences.

## EXAMPLE PRIVACY CHARACTERIZATION VALUES

[00275] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: not private/private, public/not public, and public/private.

[00276] Using no privacy and fully private as the scale endpoints, the following table lists an example privacy characterization scale.

No privacy is needed for input or output interaction The UI is not restricted to any particular I/O device for presentation and interaction. For example, the UI could present content to the user through speakers on a large monitor in a busy office.

The input must be semi-private. The output does not need to be private.

Coded speech commands, and keyboard methods are appropriate. No restrictions on output presentation.

The input must be fully private. The output does not need to be private.

No speech commands. No restriction on output presentation.

The input must be fully private. The output must be semi-private. No speech commands. No LCD panel.

The input does not need to be private. The output must be fully private.

No restrictions on input interaction. The output is restricted to an HMD device and/or an earphone.

The input does not need to be private. The output must be semi-private.

No restrictions on input interaction. The output is restricted to HMD device, earphone, and/or an LCD panel.

The input must be semi-private. The output must be semi-private. Coded speech commands and keyboard methods are appropriate. Output is restricted to an HMD device, earphone or an LCD panel.

The input and output interaction must be fully private. No speech commands. Keyboard devices might be acceptable. Output is restricted to and HMD device and/or an earphone.

[00277]       ·       Semi-private. The user and at least one other person can have access to or knowledge of the interaction between the user and the computing system.

[00278]       ·       Fully private. Only the user can have access to or knowledge of the interaction between the user and the computing system.

### COMPUTING HARDWARE CAPABILITY

[00279]       For purposes of user interfaces designs, there are four categories of hardware:

[00280]       ·       Input/output devices

[00281]       ·       Storage (e.g. RAM)

[00282]       ·       Processing capabilities

[00283]       ·       Power supply

[00284]       The hardware discussed in this topic can be the hardware that is always available to the computing system. This type of hardware is usually local to the user. Or the hardware could sometimes be available to the computing system. When a computing system uses resources that are sometimes available to it, this can be called an opportunistic use of resources.

### I/O DEVICES

[00285]       Scales for input and output devices are described later in this document.

### STORAGE

[00286]       Storage capacity refers to how much random access memory (RAM) and/or other storage is available to the computing system at any given moment. This number is not considered to be constant because the computing system might avail itself to the opportunistic use of memory.



[00287] Usually the user does not need to be aware of how much storage is available unless they are engaged in a task that might require more memory than to which they reliably have access. This might happen when the computing system engages in opportunistic use of memory. For example, if an in-motion user is engaged in a task that requires the opportunistic use of memory and that user decides to change location (e.g. they are moving from their vehicle to a utility pole where they must complete other tasks), the UI might warn the user that if they leave the current location, the computing system may not be able to complete the task or the task might not get completed as quickly.

#### EXAMPLE STORAGE CHARACTERIZATION VALUES

[00288] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no RAM is available/all RAM is available.

[00289] Using no RAM is available and all RAM is available, the following table lists an example storage characterization scale.

No RAM is available to the computing system If no RAM is available, there is no UI available.—Or—There is no change to the UI.

Of the RAM available to the computing system, only the opportunistic use of RAM is available. The UI is restricted to the opportunistic use of RAM.

Of the RAM that is available to the computing system, only the local RAM is accessible The UI is restricted to using local RAM.

Of the RAM that is available to the computing system, the RAM local to the computing system and a portion of the opportunistic use of RAM is available.

Of the RAM that is available to the computing system, the local RAM is available and the user is about to lose opportunistic use of RAM. The UI might warn the user that if they lose opportunistic use of memory, the computing system

might not be able to complete the task, or the task might not be completed as quickly.

Of the total possible RAM available to the computing system, all of it is available. If there is enough memory available to the computing system to fully function at a high level, the UI may not need to inform the user. If the user indicates to the computing system that they want a task completed that requires more memory, the UI might suggest that the user change locations to take advantage of additional opportunistic use of memory.

### PROCESSING CAPABILITIES

[00290] Processing capabilities fall into two general categories:

[00291] Speed. The processing speed of a computing system is measured in megahertz (MHz). Processing speed can be reflected as the rate of logic calculation and the rate of content delivery. The more processing power a computing system has, the faster it can calculate logic and deliver content to the user.

[00292] CPU usage. The degree of CPU usage does not affect the UI explicitly. With current UI design, if the CPU becomes too busy, the UI Typically “freezes” and the user is unable to interact with the computing system. If the CPU usage is too high, the UI will change to accommodate the CPU capabilities. For example, if the processor cannot handle the demands, the UI can simplify to reduce demand on the processor.

### EXAMPLE PROCESSING CAPABILITY

#### CHARACTERIZATION VALUES

[00293] This UI characterization is scalar, with the minimum range being binary. Example binary or scale endpoints are: no processing capability is available/all processing capability is available.

[00294] Using no processing capability is available and all processing capability as scale endpoints, the following table lists an example processing capability scale.

No processing power is available to the computing system There is no change to the UI.

The computing system has access to a slower speed CPU. The UI might be audio or text only.

The computing system has access to a high speed CPU The UI might choose to use video in the presentation instead of a still picture.

The computing system has access to and control of all processing power available to the computing system. There are no restrictions on the UI based on processing power.

#### POWER SUPPLY

[00295] There are two types of power supplies available to computing systems: alternating current (AC) and direct current (DC). Specific scale attributes for AC power supplies are represented by the extremes of the exemplary scale. However, if a user is connected to an AC power supply, it may be useful for the UI to warn an in-motion user when they're leaving an AC power supply. The user might need to switch to a DC power supply if they wish to continue interacting with the computing system while in motion. However, the switch from AC to DC power should be an automatic function of the computing system and not a function of the UI.

[00296] On the other hand, many computing devices, such as WPCs, laptops, and PDAs, operate using a battery to enable the user to be mobile. As the battery power wanes, the UI might suggest the elimination of video presentations to extend battery life. Or the UI could display a VU meter that visually demonstrates the available battery power so the user can implement their preferences accordingly.

## EXAMPLE POWER SUPPLY CHARACTERIZATION VALUES

[00297] This task characterization is binary if the power supply is AC and scalar if the power supply is DC. Example binary values are: no power/full power. Example scale endpoints are: no power/all power.

[00298] Using no power and full power as scale endpoints, the following tables lists an example power supply scale.

There is no power to the computing system. No changes to the UI are possible

There is an imminent exhaustion of power to the computing system.

The UI might suggest that the user power down the computing system before critical data is lost, or system could write most significant/useful data to display that does not require power

There is an inadequate supply of power to the computing system. If a user is listening to music, the UI might suggest that the user stop entertainment uses of the system to preserve the power supply of the computing system for critical tasks.

There is a limited, but potentially inadequate supply of power to the computing system. If the battery life is 6 hours and the computing system logic determines that the user will be away from a power source for more than 6 hours, the UI might suggest that the user conserve battery power. Or the UI might automatically operate in a “conserve power mode,” by showing still pictures instead of video or using audio instead of a visual display when appropriate.

There is a limited but adequate power supply to the computing system.

The UI might alert the user about how many hours are available in the power supply.

There is an unlimited supply of power to the computing system. The UI can use any device for presentation and interaction without restriction.

### EXEMPLARY UI DESIGN IMPLEMENTATIONS

[00299] The following list contains examples of UI design implementations for how the computing system might respond to a change in the power supply capacity.

· If there is minimal power remaining in a battery that is supporting a computing system, the UI might:

- o Power down any visual presentation surfaces, such as an LCD.
- o Use audio output only.

· If there is minimal power remaining in a battery and the UI is already audio-only, the UI might:

- o Decrease the audio output volume.
- o Decrease the number of speakers that receive the audio output or use earplugs only.

- o Use mono versus stereo output.
- o Decrease the number of confirmations to the user.

· If there is, for example, six hours of maximum-use battery life available and the computing system determines that the user not have access to a different power source for 8 hours, the UI might:

- o Decrease the luminosity of any visual display by displaying line drawings instead of 3-dimensional illustrations.
- o Change the chrominance from color to black and white.
- o Refresh the visual display less often.
- o Decrease the number of confirmations to the user.
- o Use audio output only.
- o Decrease the audio output volume.

## COMPUTING HARDWARE CHARACTERISTICS

[00300] The following is a list of some of the other hardware characteristics that may be influence what is an optimal UI design.

- Cost
- Waterproof
- Ruggedness
- Mobility

[00301] Again, there are other characteristics that could be added to this list. However, it is not possible to list all computing hardware attributes that might influence what is considered to be an optimal UI design until run time.

## INPUT/OUTPUT DEVICES

[00302] Different presentation and manipulation technologies typically have different maximum usable information densities.

## VISUAL

[00303] Visual output refers to the available visual density of the display surface is characterized by the amount of content a presentation surface can present to a user. For example, an LED output device, desktop monitor, dashboard display, hand-held device, and head mounted display all have different amounts of visual density. UI designs that are appropriate for a desktop monitor are very different than those that are appropriate for head-mounted displays. In short, what is considered to be the optimal UI will change based on what visual output device(s) is available.

[00304] In addition to density, visual display surfaces have the following characteristics:

- Color. This characterizes whether or not the presentation surface displays color. Color can be directly related to the ability of the presentation surface, of it could be assigned as a user preference.

- Chrominance. The color information in a video signal. See luminance for an explanation of chrominance and luminance.
- Motion. This characterizes whether or not a presentation surface presents motion to the user.
- Field of view. A presentation surface can display content in the focus of a user's vision, in the user's periphery, or both.
- Depth. A presentation surface can display content in 2 dimensions (e.g. a desktop monitor) or 3 dimensions (a holographic projection).
- Luminance. The amount of brightness, measured in lumens, which is given off by a pixel or area on a screen. It is the black/gray/white information in a video signal. Color information is transmitted as luminance (brightness) and chrominance (color). For example, dark red and bright red would have the same chrominance, but a different luminance. Bright red and bright green could have the same luminance, but would always have a different chrominance.
- Reflectivity. The fraction of the total radiant flux incident upon a surface that is reflected and that varies according to the wavelength distribution of the incident radiation.
- Size. Refers to the actual size of the visual presentation surface.
- Position/location of visual display surface in relation to the user and the task that they're performing.
- Number of focal points. A UI can have more than one focal point and each focal point can display different information.
- Distance of focal points from the user. A focal point can be near the user or it can be far away. The amount distance can help dictate what kind and how much information is presented to the user.
- Location of focal points in relation to the user. A focal point can be to the left of the user's vision, to the right, up, or down.

- With which eye(s) the output is associated. Output can be associated with a specific eye or both eyes.
- Ambient light.
- Others

## EXAMPLE VISUAL DENSITY CHARACTERIZATION VALUES

[00305] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no visual density/full visual density.

[00306] Using no visual density and full visual density as scale endpoints, the following table lists an example visual density scale.

There is no visual density The UI is restricted to non-visual output such as audio, haptic, and chemical.

Visual density is very low The UI is restricted to a very simple output, such as single binary output devices (a single LED) or other simple configurations and arrays of light. No text is possible.

Visual density is low The UI can handle text, but is restricted to simple prompts or the bouncing ball.

Visual density is medium The UI can display text, simple prompts or the bouncing ball, and very simple graphics.

Visual density is high The visual display has fewer restrictions. Visually dense items such as windows, icons, menus, and prompts are available as well as streaming video, detailed graphics and so on.

Visual density is very high

Visual density is the highest available The UI is not restricted by visual density. A visual display that mirrors reality (e.g. 3-dimensional) is possible and appropriate.



### EXAMPLE COLOR CHARACTERIZATION VALUES

[00307] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no color/full color.

[00308] Using no color and full color as scale endpoints, the following table lists an example color scale.

No color is available.	The UI visual presentation is monochrome.
------------------------	---

One color is available.	The UI visual presentation is monochrome plus one color.
-------------------------	--

Two colors are available	The UI visual presentation is monochrome plus two colors or any combination of the two colors.
--------------------------	--

Full color is available.	The UI is not restricted by color.
--------------------------	------------------------------------

### EXAMPLE MOTION CHARACTERIZATION VALUES

[00309] This UI characterization is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: no motion is available/full motion is available.

[00310] Using no motion is available and full motion is available as scale endpoints, the following table lists an example motion scale.

No motion is available	The UI is restricted by motion. There are no videos, streaming videos moving text, and so on.
------------------------	---

Limited motion is available
-----------------------------

Moderate motion is available
------------------------------

Full range of motion is available	The UI is not restricted by motion.
-----------------------------------	-------------------------------------

### EXAMPLE FIELD OF VIEW CHARACTERIZATION VALUES

[00311] This UI characterization is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are : peripheral vision only/field of focus and peripheral vision is available.

[00312] Using peripheral vision only and field of focus and peripheral vision is available as scale endpoints, the following tables lists an example field of view scale.

All visual display is in the peripheral vision of the userThe UI is restricted to using the peripheral vision of the user. Lights, colors and other simple visual display are appropriate. Text is not appropriate.

Only the user's field of focus is available. The UI is restricted to using the users field of vision only. Text and other complex visual displays are appropriate.

Both field of focus and the peripheral vision of the user are used. The UI is not restricted by the user's field of view.

#### EXEMPLARY UI DESIGN IMPLEMENTATION FOR CHANGES IN FIELD OF VIEW

[00313] The following list contains examples of UI design implementations for how the computing system might respond to a change in field of view.

If the field of view for the visual presentation is more than 28°, then the UI might:

- o Display the most important information at the center of the visual presentation surface.

- o Devote more of the UI to text

- o Use periphicons outside of the field of view.

If the field of view for the visual presentation is less than 28°, then the UI might:

- o Restrict the size of the font allowed in the visual presentation. For example, instead of listing "Monday, Tuesday, and Wednesday," and so on as choices, the UI might list "M, Tu, W" instead.

- o The body or environment stabilized image can scroll.

## EXAMPLE DEPTH CHARACTERIZATION VALUES

[00314] This characterization is binary and the values are: 2 dimensions, 3 dimensions.

[00315] Exemplary UI design implementation for changes in reflectivity

[00316] The following list contains examples of UI design implementations for how the computing system might respond to a change in reflectivity.

[00317] If the output device has high reflectivity — a lot of glare — then the visual presentation will change to a light colored UI.

### AUDIO

[00318] Audio input and output refers to the UI's ability to present and receive audio signals. While the UI might be able to present or receive any audio signal strength, if the audio signal is outside the human hearing range (approximately 20 Hz to 20,000 Hz) it is converted so that it is within the human hearing range, or it is transformed into a different presentation, such as haptic output, to provide feedback, status, and so on to the user.

[00319] Factors that influence audio input and output include (but this is not an inclusive list):

- Level of ambient noise (this is an environmental characterization)
- Directionality of the audio signal
- Head stabilized output (e.g. earphones)
- Environment stabilized output (e.g. speakers)
- Spatial layout (3-D audio)
- Proximity of the audio signal to the user
- Frequency range of the speaker
- Fidelity of the speaker, e.g. total harmonic distortion
- Left, right, or both ears
- What kind of noise is it?

Others

## EXAMPLE AUDIO OUTPUT CHARACTERIZATION VALUES

[00320] This characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: the user cannot hear the computing system/ the user can hear the computing system.

[00321] Using the user cannot hear the computing system and the user can hear the computing system as scale endpoints, the following table lists an example audio output characterization scale.

The user cannot hear the computing system. The UI cannot use audio to give the user choices, feedback, and so on.

The user can hear audible whispers (approximately 10-30 dBA). The UI might offer the user choices, feedback, and so on by using the earphone only.

The user can hear normal conversation (approximately 50-60 dBA).

The UI might offer the user choices, feedback, and so on by using a speaker(s) connected to the computing system.

The user can hear communications from the computing system without restrictions. The UI is not restricted by audio signal strength needs or concerns.

Possible ear damage (approximately 85+ dBA) The UI will not output audio for extended periods of time that will damage the user's hearing.

## EXAMPLE AUDIO INPUT CHARACTERIZATION VALUES

[00322] This characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: the computing system cannot hear the user/the computing system can hear the user.

[00323] Using the computing system cannot hear the user and the computing system can hear the user as scale endpoints, the following table lists an example audio input scale.

The computing system cannot receive audio input from the user. When the computing system cannot receive audio input from the user, the UI will notify the user that audio input is not available.

The computing system is able to receive audible whispers from the user (approximate 10-30 dBA).

The computing system is able to receive normal conversational tones from the user (approximate 50-60 dBA).

The computing system can receive audio input from the user without restrictions. The UI is not restricted by audio signal strength needs or concerns.

The computing system can receive only high volume audio input from the user. The computing system will not require the user to give indications using a high volume. If a high volume is required, then the UI will notify the user that the computing system cannot receive audio input from the user.

## HAPTICS

[00324] Haptics refers to interacting with the computing system using a tactile method. Haptic input includes the computing system's ability to sense the user's body movement, such as finger or head movement. Haptic output can include applying pressure to the user's skin. For haptic output, the more transducers, the more skin covered, the more resolution for presentation of information. That is if the user is covered with transducers, the computing system receives a lot more input from the user. Additionally, the ability for haptically-oriented output presentations is far more flexible.

## EXAMPLE HAPTIC INPUT CHARACTERIZATION VALUES

[00325] This characteristic is enumerated. Possible values include accuracy, precision, and range of:

- Pressure
- Velocity

- Temperature
- Acceleration
- Torque
- Tension
- Distance
- Electrical resistance
- Texture
- Elasticity
- Wetness

[00326] Additionally, the characteristics listed previously are enhanced by:

- Number of dimensions
- Density and quantity of sensors (e.g. a 2 dimensional array of

sensors. The sensors could measure the characteristics previously listed).

#### CHEMICAL OUTPUT

[00327] Chemical output refers to using chemicals to present feedback, status, and so on to the user. Chemical output can include:

- Things a user can taste
- Things a user can smell

[00328] Characteristics of taste include:

- Bitter
- Sweet
- Salty
- Sour

[00329] Characteristics of smell include:

- Strong/weak
- Pungent/bland
- Pleasant/unpleasant

· Intrinsic, or signaling

### ELECTRICAL INPUT

[00330] Electrical input refers to a user's ability to actively control electrical impulses to send indications to the computing system.

- Brain activity
- Muscle activity

[00331] Characteristics of electrical input can include:

- Strength of impulse

### BANDWIDTH

[00332] There are different types of bandwidth, for instance:

- Network bandwidth
- Inter-device bandwidth

### NETWORK BANDWIDTH

[00333] Network bandwidth is the computing system's ability to connect to other computing resources such as servers, computers, printers, and so on. A network can be a local area network (LAN), wide area network (WAN), peer-to-peer, and so on. For example, if the user's preferences are stored at a remote location and the computing system determines that the remote resources will not always be available, the system might cache the user's preferences locally to keep the UI consistent. As the cache may consume some of the available RAM resources, the UI might be restricted to simpler presentations, such as text or audio only.

[00334] If user preferences cannot be cached, then the UI might offer the user choices about what UI design families are available and the user can indicate their design family preference to the computing system.

## EXAMPLE NETWORK BANDWIDTH CHARACTERIZATION VALUES

[00335] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no network access/full network access.

[00336] Using no network access and full network access as scale endpoints, the following table lists an example network bandwidth scale.

The computing system does not have a connection to network resources.

The UI is restricted to using local computing resources only. If user preferences are stored remotely, then the UI might not account for user preferences.

The computing system has an unstable connection to network resources.

The UI might warn the user that the connection to remote resources might be interrupted. The UI might ask the user if they want to cache appropriate information to accommodate for the unstable connection to network resources.

The computing system has a slow connection to network resources.

The UI might simplify, such as offer audio or text only, to accommodate for the slow connection. Or the computing system might cache appropriate data for the UI so the UI can always be optimized without restriction of the slow connection.

The computing system has a high speed, yet limited (by time) access to network resources. In the present moment, the UI does not have any restrictions based on access to network resources. If the computing system determines that it will lose a network connection, then the UI can warn the user and offer choices, such as does the user want to cache appropriate information, about what to do.



The computing system has a very high-speed connection to network resources. There are no restrictions to the UI based on access to network resources. The UI can offer text, audio, video, haptic output, and so on.

### INTER-DEVICE BANDWIDTH

[00337] Inter-device bandwidth is the ability of the devices that are local to the user to communicate with each other. Inter-device bandwidth can affect the UI in that if there is low inter-device bandwidth, then the computing system cannot compute logic and deliver content as quickly. Therefore, the UI design might be restricted to a simpler interaction and presentation, such as audio or text only. If bandwidth is optimal, then there are no restrictions on the UI based on bandwidth. For example, the UI might offer text, audio, and 3-D moving graphics if appropriate for the user's context.

### EXAMPLE INTER-DEVICE BANDWIDTH CHARACTERIZATION VALUES

[00338] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no inter-device bandwidth/full inter-device bandwidth.

[00339] Using no inter-device bandwidth and full inter-device bandwidth as scale endpoints, the following table lists an example inter-device bandwidth scale.

The computing system does not have inter-device connectivity. Input and output is restricted to each of the disconnected devices. The UI is restricted to the capability of each device as a stand-alone device.

Some devices have connectivity and others do not. It depends

The computing system has slow inter-device bandwidth. The task that the user wants to complete might require more bandwidth that is available among devices. In this case, the UI can offer the user a choice. Does the user want to continue and encounter slow performance? Or, does the user want to acquire

more bandwidth by moving to a different location and taking advantage of opportunistic use of bandwidth?

The computing system has fast inter-device bandwidth. There are few, if any, restrictions on the interaction and presentation between the user and the computing system. The UI sends a warning message only if there is not enough bandwidth between devices.

The computing system has very high-speed inter-device connectivity.

There are no restrictions on the UI based on inter-device connectivity.

## EXPOSING DEVICE CHARACTERIZATION TO THE COMPUTING SYSTEM

[00340] There are many ways to expose the context characterization to the computing system, as shown by the following three examples.

### NUMERIC KEY

[00341] A context characterization can be exposed to the system with a numeric value corresponding to values of a predefined data structure.

[00342] For instance, a binary number can have each of the bit positions associated with a specific characteristic. The least significant bit may represent the need for a visual display device capable of displaying at least 24 characters of text in an unbroken series. Therefore a UI characterization of decimal 5 would require such a display to optimally display its content.

### XML TAGS

[00343] A UI's characterization can be exposed to the system with a string of characters conforming to the XML structure.

[00344] For instance, a context characterization might be represented by the following:

```
<Context Characterization>
<Theme>Work </Theme>
```

<Bandwidth>High Speed LAN Network Connection</Bandwidth>

<Field of View> 28° </Field of View>

<Privacy> None </Privacy>

</Context Characterization>

[00345] One significant advantage of the mechanism is that it is easily extensible.

#### PROGRAMMING INTERFACE

[00346] A context characterization can be exposed to the computing system by associating the design with a specific program call.

[00347] For instance:

GetSecureContext can return a handle to the computing system that describes a UI a high security user context.

#### NAME/VALUE PAIRS

[00348] A context is modeled or represented with multiple attributes that each correspond to a specific element of the context (e.g., ambient temperature, location or a current user activity), and the value of an attribute represents a specific measure of that element. Thus, for example, for an attribute that represents the temperature of the surrounding air, an 80° Fahrenheit value represents a specific measurement of that temperature. Each attribute preferably has the following properties: a name, a value, an uncertainty level, units, and a timestamp. Thus, for example, the name of the air temperature attribute may be "ambient-temperature," its units may be degrees Fahrenheit, and its value at a particular time may be 80. Associated with the current value may be a timestamp of 02/27/99 13:07 PST that indicates when the value was generated, and an uncertainty level of +/- 1 degrees.

## DETERMINING UI REQUIREMENTS FOR AN OPTIMAL OR APPROPRIATE UI

[00349] Considered singularly, many of the characteristics described below can be beneficially used to inform a computing system when to change. However, with an extensible system, additional characteristics can be considered (or ignored) at anytime, providing precision to the optimization.

### ATTRIBUTES ANALYZED

[00350] At least the following categories of attributes can be used when determining the optimal UI design:

[00351] \* All available attributes. The model is dynamic so it can accommodate for any and all attributes that could affect the optimal UI design for a user's context. For example, this model could accommodate for temperature, weather conditions, time of day, available I/O devices, preferred volume level, desired level of privacy, and so on.

[00352] \* Significant attributes. Some attributes have a more significant influence on the optimal UI design than others. Significant attributes include, but are not limited to, the following:

- \* The user can see video.
- \* The user can hear audio.
- \* The computing system can hear the user.
- \* The interaction between the user and the computing system must be private.
- \* The user's hands are occupied.

[00353] \* Attributes that correspond to a theme. Specific or programmatic. Individual or group.

[00354] The attributes discussed below are meant to be illustrative because it is often not possible to know all of the attributes that will affect a UI design until run time. Thus, the described techniques are dynamic to allowing accounting for

unknown attributes. For clarity, attributes described below are presented with a scale and some include design examples. It is important to note that any of the attributes mentioned in this document are just examples. There are other attributes that can cause a UI to change that are not listed in this document. However, the dynamic model can account for additional attributes.

### I/O DEVICES

[00355] Output — Devices that are directly perceivable by the user. For example, a visual output device creates photons that enter the user's eye. Output devices are always local to the user.

[00356] Input — A device that can be directly manipulated by the user. For example, a microphone translates energy created by the user's voice into electrical signals that can control a computer. Input devices are always local to the user.

[00357] The input devices to which the user has access to interact with the computer in ways that convey choices include, but are not limited to:

- \* Keyboards
- \* Touch pads
- \* Mice
- \* Trackballs
- \* Microphones
- \*

Rolling/pointing/pressing/bending/turning/twisting/switching/rubbing/zippping  
cursor controllers - anything that the user's manipulation of can be sensed by the computer, this includes body movement that forms recognizable gestures.

- \* Buttons, etc.

[00358] Output devices allow the presentation of computer-controlled information and content to the user, and include:

- \* Speakers
- \* Monitors
- \* Pressure actuators, etc.

### INPUT DEVICE TYPES

[00359] Some characterizations of input devices are a direct result of the device itself.

### TOUCH SCREEN

[00360] A display screen that is sensitive to the touch of a finger or stylus. Touch screens are very resistant to harsh environments where keyboards might eventually fail. They are often used with custom-designed applications so that the on-screen buttons are large enough to be pressed with the finger. Applications are typically very specialized and greatly simplified so they can be used by anyone. However, touch screens are also very popular on PDAs and full-size computers with standard applications, where a stylus is required for precise interaction with screen objects.

### EXAMPLE TOUCH SCREEN ATTRIBUTE CHARACTERISTIC VALUES

[00361] This characteristic is enumerated. Some example values are:

- \* Screen objects must be at least 1 centimeter square
- \* The user can see the touch screen directly
- \* The user can see the touch screen indirectly (e.g. by using a monitor)
- \* Audio feedback is available
- \* Spatial input is difficult
- \* Feedback to the user is presented to the user through a visual presentation surface.

### POINTING DEVICE

[00362] An input device used to move the pointer (cursor) on screen.

## EXAMPLE POINTING DEVICE CHARACTERISTIC VALUES

[00363] This characteristic is enumerated. Some example values are:

- \* 1-dimension (D) pointing device
- \* 2-D pointing device
- \* 3-D pointing device
- \* Position control device
- \* Range control device
- \* Feedback to the user is presented through a visual presentation

surface.

## SPEECH

[00364] The conversion of spoken words into computer text. Speech is first digitized and then matched against a dictionary of coded waveforms. The matches are converted into text as if the words were typed on the keyboard.

## EXAMPLE SPEECH CHARACTERISTIC VALUES

[00365] This characteristic is enumerated. Example values are:

- \* Command and control
- \* Dictation
- \* Constrained grammar
- \* Unconstrained grammar

## KEYBOARD

[00366] A set of input keys. On terminals and personal computers, it includes the standard typewriter keys, several specialized keys and the features outlined below.

## EXAMPLE KEYBOARD CHARACTERISTIC VALUES

[00367] This characteristic is enumerated. Example values are:

- \* Numeric

- \* Alphanumeric
- \* Optimized for discreet input

#### PEN TABLET

[00368] A digitizer tablet that is specialized for handwriting and hand marking. LCD-based tablets emulate the flow of ink as the tip touches the surface and pressure is applied. Non-display tablets display the handwriting on a separate computer screen.

#### EXAMPLE PEN TABLET CHARACTERISTIC VALUES

[00369] This characteristic is enumerated. Example values include:

- \* Direct manipulation device
- \* Feedback is presented to the user through a visual presentation surface
- \* Supplemental feedback can be presented to the user using audio output.
- \* Optimized for special input
- \* Optimized for data entry

#### EYE TRACKING

[00370] An eye-tracking device is a device that uses eye movement to send user indications about choices to the computing system. Eye tracking devices are well suited for situations where there is little to no motion from the user (e.g. the user is sitting at a desk) and has much potential for non-command user interfaces.

#### EXAMPLE EYE TRACKING CHARACTERISTIC VALUES

[00371] This characteristic is enumerated. Example values include:

- \* 2-D pointing device
- \* User motion = still
- \* Privacy = high



## OUTPUT DEVICE TYPES

- [00372] Some characterizations of input devices are a direct result of the device itself.

### HMD

- [00373] Head Mounted Display) A display system built and worn like goggles that gives the illusion of a floating monitor in front of the user's face. The HMD is an important component of a body-worn computer (wearable computer). Single-eye units are used to display hands-free instructional material, and dual-eye, or stereoscopic, units are used for virtual reality applications.

### EXAMPLE HMD CHARACTERISTIC VALUES

- [00374] This characteristic is enumerated. Example values include:
- \* Field of view > 28°
  - \* User's hands = not available
  - \* User's eyes = forward and out
  - \* User's reality = augmented, mediated, or virtual

### MONITORS

- [00375] A display screen used to present output from a computer, camera, VCR or other video generator. A monitor's clarity is based on video bandwidth, dot pitch, refresh rate, and convergence.

### EXAMPLE MONITOR CHARACTERISTIC VALUES

- [00376] This characteristic is enumerated. Some example values include:
- \* Required graphical resolution = high
  - \* User location = stationary
  - \* User attention = high
  - \* Visual density = high
  - \* Animation = yes
  - \* Simultaneous presentation of information = yes (e.g. text and image)

- \* Spatial content = yes

#### I/O DEVICE USE

[00377] This attribute characterizes how or for what an input or output device can be optimized for use. For example, a keyboard is optimized for entering alphanumeric text characters and monitor, head mounted display (HMD), or LCD panel is optimized for displaying those characters and other visual information.

#### EXAMPLE DEVICE USE CHARACTERIZATION VALUES

[00378] This characterization is enumerated. Example values include:

- \* Speech recognition
- \* Alphanumeric character input
- \* Handwriting recognition
- \* Visual presentation
- \* Audio presentation
- \* Haptic presentation
- \* Chemical presentation

#### REDUNDANT CONTROLS

[00379] The user may have more than one way to perceive or manipulate the computing environment. For instance, they may be able to indicate choices by manipulating a mouse, or speaking.

[00380] By providing UI designs that have more than one I/O modality (also known as “multi-modal”), greater flexibility can be provided to the user. However, there are times when this is not appropriate. For instance, the devices may not be constantly available (user’s hands are occupied, the ambient noise increases defeating voice recognition).

## EXAMPLE REDUNDANT CONTROLS CHARACTERIZATION VALUES

[00381] As a minimum, a numeric value could be associated with a configuration of devices.

- \* 1 – keyboard and touch screen
- \* 2 – HMD and 2-D pointing device

[00382] Alternately, a standardized list of available, preferred, or historically used devices could be used.

- \* QWERTY keyboard
- \* Twiddler
- \* HMD
- \* VGA monitor
- \* SVGA monitor
- \* LCD display
- \* LCD panel

## PRIVACY

[00383] Privacy is the quality or state of being apart from company or observation. It includes a user's trust of audience. For example, if a user doesn't want anyone to know that they are interacting with a computing system (such as a wearable computer), the preferred output device might be an HMD and the preferred input device might be an eye-tracking device.

## HARDWARE AFFINITY FOR PRIVACY

[00384] Some hardware suits private interactions with a computing system more than others. For example, an HMD is a far more private output device than a desk monitor. Similarly, an earphone is more private than a speaker.

[00385] The UI should choose the correct input and output devices that are appropriate for the desired level of privacy for the user's current context and preferences.

#### EXAMPLE PRIVACY CHARACTERIZATION VALUES

[00386] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: not private/private, public/not public, and public/private.

[00387] Using no privacy and fully private as the scale endpoints, the following table lists an example privacy characterization scale.

Scale attribute	Implication/Example
No privacy is needed for input or output interaction	The UI is not restricted to any particular I/O device for presentation and interaction. For example, the UI could present content to the user through speakers on a large monitor in a busy office.
The input must be semi-private. The output does not need to be private.	Coded speech commands, and keyboard methods are appropriate. No restrictions on output presentation.
The input must be fully private. The output does not need to be private.	No speech commands. No restriction on output presentation.
The input must be fully private. The output must be semi-private.	No speech commands. No LCD panel.
The input does not need to be private. The output must be fully private.	No restrictions on input interaction. The output is restricted to an HMD device and/or an earphone.

Scale attribute	Implication/Example
The input does not need to be private. The output must be semi-private.	No restrictions on input interaction. The output is restricted to HMD device, earphone, and/or an LCD panel.
The input must be semi-private. The output must be semi-private.	Coded speech commands and keyboard methods are appropriate. Output is restricted to an HMD device, earphone or an LCD panel.
The input and output interaction must be fully private.	No speech commands. Keyboard devices might be acceptable. Output is restricted to and HMD device and/or an earphone.

[00406] \* Semi-private. The user and at least one other person can have access to or knowledge of the interaction between the user and the computing system.

[00407] \* Fully private. Only the user can have access to or knowledge of the interaction between the user and the computing system.

## VISUAL

[00408] Visual output refers to the available visual density of the display surface is characterized by the amount of content a presentation surface can present to a user. For example, an LED output device, desktop monitor, dashboard display, hand-held device, and head mounted display all have different amounts of visual density. UI designs that are appropriate for a desktop monitor are very different than those that are appropriate for head-mounted displays. In short, what is considered to be the optimal UI will change based on what visual output device(s) is available.

[00409] In addition to density, visual display surfaces have the following characteristics:

- \* Color
- \* Motion
- \* Field of view
- \* Depth
- \* Reflectivity
- \* Size. Refers to the actual size of the visual presentation surface.
- \* Position/location of visual display surface in relation to the user and the task that they're performing.
- \* Number of focal points. A UI can have more than one focal point and each focal point can display different information.
- \* Distance of focal points from the user. A focal point can be near the user or it can be far away. The amount distance can help dictate what kind and how much information is presented to the user.
- \* Location of focal points in relation to the user. A focal point can be to the left of the user's vision, to the right, up, or down.
- \* With which eye(s) the output is associated. Output can be associated with a specific eye or both eyes.
- \* Ambient light.
- \* Others (e.g., cost, flexibility, breakability, mobility, exit pupil, . . .)

[00410] The topics in this section describe in further detail the characteristics of some of these previously listed attributes.

#### EXAMPLE VISUAL DENSITY CHARACTERIZATION VALUES

[00411] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no visual density/full visual density.

[00412] Using no visual density and full visual density as scale endpoints, the following table lists an example visual density scale. Note that in some situations density might not be uniform across the presentation surface. For example, it may mimic the eye and have high resolution toward the center where text could be supported, but low resolution at the periphery where graphics are appropriate.

Scale attribute	Implication/Design example
There is no visual density	The UI is restricted to non-visual output such as audio, haptic, and chemical.
Visual density is very low	The UI is restricted to a very simple output, such as single binary output devices (a single LED) or other simple configurations and arrays of light. No text is possible.
Visual density is low	The UI can handle text, but is restricted to simple prompts or the bouncing ball.
Visual density is medium	The UI can display text, simple prompts or the bouncing ball, and very simple graphics.
Visual density is high	The visual display has fewer restrictions. Visually dense items such as windows, icons, menus, and prompts are available as well as streaming video, detailed graphics and so on.
Visual density is the highest available	The UI is not restricted by visual density. A visual display that mirrors

Scale attribute	Implication/Design example
	reality (e.g. 3-dimensional) is possible and appropriate.

## COLOR

[00427] This characterizes whether or not the presentation surface displays color. Color can be directly related to the ability of the presentation surface, or it could be assigned as a user preference.

\* Chrominance. The color information in a video signal.

\* Luminance. The amount of brightness, measured in lumens, which is given off by a pixel or area on a screen. It is the black/gray/white information in a video signal. Color information is transmitted as luminance (brightness) and chrominance (color). For example, dark red and bright red would have the same chrominance, but a different luminance. Bright red and bright green could have the same luminance, but would always have a different chrominance.

### EXAMPLE COLOR CHARACTERIZATION VALUES

[00428] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no color/full color.

[00429] Using no color and full color as scale endpoints, the following table lists an example color scale.

Scale attribute	Implication/Design example
No color is available.	The UI visual presentation is monochrome.
One color is available.	The UI visual presentation is monochrome plus one color.
Two colors are available	The UI visual presentation is monochrome plus two colors or any



Scale attribute	Implication/Design example
	combination of the two colors.
Full color is available.	The UI is not restricted by color.

## MOTION

- [00440] This characterizes whether or not a presentation surface has the ability to present motion to the user. Motion can be considered as a stand-alone attribute or as a composite attribute.

### EXAMPLE MOTION CHARACTERIZATION VALUES

- [00441] As a stand-alone attribute, this characterization is binary. Example binary values are: no animation available/animation available.
- [00442] As a composite attribute, this characterization is scalar. Example scale endpoints include no motion/motion available, no animation available/animation available, or no video/video. The values between the endpoints depend on the other characterizations that are included in the composite. For example, the attributes color, visual density, and frames per second, etc. change the values between no motion and motion available.

## FIELD OF VIEW

- [00443] A presentation surface can display content in the focus of a user's vision, in the user's periphery, or both.

### EXAMPLE FIELD OF VIEW CHARACTERIZATION VALUES

- [00444] This UI characterization is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: peripheral vision only/field of focus and peripheral vision is available.

[00445] Using peripheral vision only and field of focus and peripheral vision is available as scale endpoints, the following tables lists an example field of view scale.

Scale attribute	Implication
All visual display is in the peripheral vision of the user	The UI is restricted to using the peripheral vision of the user. Lights, colors and other simple visual display are appropriate. Text is not appropriate.
Only the user's field of focus is available.	The UI is restricted to using the users field of vision only. Text and other complex visual displays are appropriate.
Both field of focus and the peripheral vision of the user are used.	The UI is not restricted by the user's field of view.

#### EXEMPLARY UI DESIGN IMPLEMENTATION FOR CHANGES IN FIELD OF VIEW

[00454] The following list contains examples of UI design implementations for how the computing system might respond to a change in field of view.

\* If the field of view for the visual presentation is more than 28°, then the UI might:

\* Display the most important information at the center of the visual presentation surface.

\* Devote more of the UI to text

\* Use periphicons outside of the field of view.

\* If the field of view for the visual presentation is less than 28°, then the UI might:

\* Restrict the size of the font allowed in the visual presentation. For example, instead of listing “Monday, Tuesday, and Wednesday,” and so on as choices, the UI might list “M, Tu, W” instead.

\* The body or environment stabilized image can scroll.

#### DEPTH

[00455] A presentation surface can display content in 2 dimensions (e.g., a desktop monitor) or 3 dimensions (a holographic projection).

#### EXAMPLE DEPTH CHARACTERIZATION VALUES

[00456] This characterization is binary and the values are: 2 dimensions, 3 dimensions.

#### REFLECTIVITY

[00457] The fraction of the total radiant flux incident upon a surface that is reflected and that varies according to the wavelength distribution of the incident radiation.

#### EXAMPLE REFLECTIVITY CHARACTERIZATION VALUES

[00458] This characterization is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: not reflective/highly reflective or no glare/high glare.

[00459] Using not reflective and highly reflective as scale endpoints, the following list is an example of a reflectivity scale.

- \* Not reflective (no surface reflectivity).

- \* 10% surface reflectivity

- \* 20% surface reflectivity

- \* 30% surface reflectivity

- \* 40% surface reflectivity

- \* 50% surface reflectivity

- \* 60% surface reflectivity

- \* 70% surface reflectivity
- \* 80% surface reflectivity
- \* 90% surface reflectivity
- \* Highly reflective (100% surface reflectivity)

#### EXEMPLARY UI DESIGN IMPLEMENTATION FOR CHANGES IN REFLECTIVITY

[00460] The following list contains examples of UI design implementations for how the computing system might respond to a change in reflectivity.

- \* If the output device has high reflectivity — a lot of glare — then the visual presentation will change to a light colored UI.

#### AUDIO

[00461] Audio input and output refers to the UI's ability to present and receive audio signals. While the UI might be able to present or receive any audio signal strength, if the audio signal is outside the human hearing range (approximately 20 Hz to 20,000 Hz) it is converted so that it is within the human hearing range, or it is transformed into a different presentation, such as haptic output, to provide feedback, status, and so on to the user

[00462] Factors that influence audio input and output include (but this is not an inclusive list):

- \* Level of ambient noise (this is an environmental characterization)
- \* Directionality of the audio signal
- \* Head-stabilized output (e.g. earphones)
- \* Environment-stabilized output (e.g. speakers)
- \* Spatial layout (3-D audio)
- \* Proximity of the audio signal to the user
- \* Frequency range of the speaker
- \* Fidelity of the speaker, e.g. total harmonic distortion

- \* Left, right, or both ears
- \* What kind of noise is it?
- \* Others (e.g., cost, proximity to other people, . . . )

#### EXAMPLE AUDIO OUTPUT CHARACTERIZATION VALUES

[00463] This characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: the user cannot hear the computing system/ the user can hear the computing system.

[00464] Using the user cannot hear the computing system and the user can hear the computing system as scale endpoints, the following table lists an example audio output characterization scale.

Scale attribute	Implication
The user cannot hear the computing system.	The UI cannot use audio to give the user choices, feedback, and so on.
The user can hear audible whispers (approximately 10-30 dBA).	The UI might offer the user choices, feedback, and so on by using the earphone only.
The user can hear normal conversation (approximately 50-60 dBA).	The UI might offer the user choices, feedback, and so on by using a speaker(s) connected to the computing system.
The user can hear communications from the computing system without restrictions.	The UI is not restricted by audio signal strength needs or concerns.
Possible ear damage (approximately 85+ dBA)	The UI will not output audio for extended periods of time that will damage the user's hearing.

## EXAMPLE AUDIO INPUT CHARACTERIZATION VALUES

[00477] This characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: the computing system cannot hear the user/the computing system can hear the user.

[00478] Using the computing system cannot hear the user and the computing system can hear the user as scale endpoints, the following table lists an example audio input scale.

Scale attribute	Implication
The computing system cannot receive audio input from the user.	When the computing system cannot receive audio input from the user, the UI will notify the user that audio input is not available.
The computing system is able to receive audible whispers from the user (approximate 10-30 dBA).	
The computing system is able to receive normal conversational tones from the user (approximate 50-60 dBA).	
The computing system can receive audio input from the user without restrictions.	The UI is not restricted by audio signal strength needs or concerns.
The computing system can receive only high volume audio input from the user.	The computing system will not require the user to give indications using a high volume. If a high volume is required, then the UI will notify the user

	that the computing system cannot receive audio input from the user.
--	---

## HAPTICS

[00491] Haptics refers to interacting with the computing system using a tactile method. Haptic input includes the computing system's ability to sense the user's body movement, such as finger or head movement. Haptic output can include applying pressure to the user's skin. For haptic output, the more transducers, the more skin covered, the more resolution for presentation of information. That is if the user is covered with transducers, the computing system receives a lot more input from the user. Additionally, the ability for haptically-oriented output presentations is far more flexible.

### EXAMPLE HAPTIC INPUT CHARACTERIZATION VALUES

[00492] This characteristic is enumerated. Possible values include accuracy, precision, and range of:

- \* Pressure
- \* Velocity
- \* Temperature
- \* Acceleration
- \* Torque
- \* Tension
- \* Distance
- \* Electrical resistance
- \* Texture
- \* Elasticity
- \* Wetness

Additionally, the characteristics listed previously are enhanced by:

\* Number of dimensions

\* Density and quantity of sensors (e.g., a 2 dimensional array of sensors.

The sensors could measure the characteristics previously listed).

### CHEMICAL OUTPUT

[00493] Chemical output refers to using chemicals to present feedback, status, and so on to the user. Chemical output can include:

\* Things a user can taste

\* Things a user can smell

### EXAMPLE TASTE CHARACTERISTIC VALUES

[00494] This characteristic is enumerated. Example characteristic values of taste include:

\* Bitter

\* Sweet

\* Salty

\* Sour

### EXAMPLE SMELL CHARACTERISTIC VALUES

[00495] This characteristic is enumerated. Example characteristic values of smell include:

\* Strong/weak

\* Pungent/bland

\* Pleasant/unpleasant

\* Intrinsic, or signaling

### ELECTRICAL INPUT

[00496] Electrical input refers to a user's ability to actively control electrical impulses to send indications to the computing system.

\* Brain activity

\* Muscle activity



## EXAMPLE ELECTRICAL INPUT CHARACTERIZATION VALUES

[00497] This characteristic is enumerated. Example values of electrical input can include:

- \* Strength of impulse
- \* Frequency

### USER CHARACTERIZATIONS

[00498] This section describes the characteristics that are related to the user.

### USER PREFERENCES

[00499] User preferences are a set of attributes that reflect the user's likes and dislikes, such as I/O devices preferences, volume of audio output, amount of haptic pressure, and font size and color for visual display surfaces. User preferences can be classified in the following categories:

[00500] \* Self characterization. Self-characterized user preferences are indications from the user to the computing system about themselves. The self-characterizations can be explicit or implicit. An explicit, self-characterized user preference results in a tangible change in the interaction and presentation of the UI. An example of an explicit, self characterized user preference is "Always use the font size 18" or "The volume is always off." An implicit, self-characterized user preference results in a change in the interaction and presentation of the UI, but it might be not be immediately tangible to the user. A learning style is an implicit self-characterization. The user's learning style could affect the UI design, but the change is not as tangible as an explicit, self-characterized user preference.

[00501] If a user characterizes themselves to a computing system as a "visually impaired, expert computer user," the UI might respond by always using 24-point font and monochrome with any visual display surface. Additionally, tasks would be

chunked differently, shortcuts would be available immediately, and other accommodations would be made to tailor the UI to the expert user.

[00502]       \* Theme selection. In some situations, it is appropriate for the computing system to change the UI based on a specific theme. For example, a high school student in public school 1420 who is attending a chemistry class could have a UI appropriate for performing chemistry experiments. Likewise, an airplane mechanic could also have a UI appropriate for repairing airplane engines. While both of these UIs would benefit from hands free, eyes out computing, the UI would be specifically and distinctively characterized for that particular system.

[00503]       \* System characterization. When a computing system somehow infers a user's preferences and uses those preferences to design an optimal UI, the user preferences are considered to be system characterizations. These types of user preferences can be analyzed by the computing system over a specified period on time in which the computing system specifically detects patterns of use, learning style, level of expertise, and so on. Or, the user can play a game with the computing system that is specifically designed to detect these same characteristics.

[00504]       \* Pre-configured. Some characterizations can be common and the UI can have a variety of pre-configured settings that the user can easily indicate to the UI. Pre-configured settings can include system settings and other popular user changes to default settings.

[00505]       \* Remotely controlled. From time to time, it may be appropriate for someone or something other than the user to control the UI that is displayed.

#### EXAMPLE USER PREFERENCE CHARACTERIZATION VALUES

[00506]       This UI characterization scale is enumerated. Some example values include:

- \* Self characterization
- \* Theme selection
- \* System characterization
- \* Pre-configured
- \* Remotely controlled

#### THEME

[00507] A theme is a related set of measures of specific context elements, such as ambient temperature, current user task, and latitude, which reflect the context of the user. In other words, theme is a name collection of attributes, attribute values, and logic that relates these things. Typically, themes are associated with user goals, activities, or preferences. The context of the user includes:

- \* The user's mental state, emotional state, and physical or health condition.
- \* The user's setting, situation or physical environment. This includes factors external to the user that can be observed and/or manipulated by the user, such as the state of the user's computing system.
- \* The user's logical and data telecommunications environment (or "cyber-environment," including information such as email addresses, nearby telecommunications access such as cell sites, wireless computer ports, etc.).

[00508] Some examples of different themes include: home, work, school, and so on. Like user preferences, themes can be self characterized, system characterized, inferred, pre-configured, or remotely controlled.

#### EXAMPLE THEME CHARACTERIZATION VALUES

[00509] This characteristic is enumerated. The following list contains example enumerated values for theme.

- \* No theme
- \* The user's theme is inferred.
- \* The user's theme is pre-configured.

- \* The user's theme is remotely controlled.
- \* The user's theme is self characterized.
- \* The user's theme is system characterized.

### USER CHARACTERISTICS

[00510] User characteristics include:

- \* Emotional state
- \* Physical state
- \* Cognitive state
- \* Social state

### EXAMPLE USER CHARACTERISTICS

#### CHARACTERIZATION VALUES

[00511] This UI characterization scale is enumerated. The following lists contain some of the enumerated values for each of the user characteristic qualities listed above.

- \* Emotional state.
  - \* Happiness
  - \* Sadness
  - \* Anger
  - \* Frustration
  - \* Confusion
- \* Physical state
  - \* Body
  - \* Biometrics
  - \* Posture
  - \* Motion
  - \* Physical Availability
    - \* Senses

- \* Eyes
- \* Ears
- \* Tactile
- \* Hands
- \* Nose
- \* Tongue
- \* Workload demands/effects
- \* Interaction with computer devices
- \* Interaction with people
- \* Physical Health
- \* Environment
  - \* Time/Space
  - \* Objects
  - \* Persons
    - \* Audience/Privacy Availability
      - \* Scope of Disclosure
      - \* Hardware affinity for privacy
      - \* Privacy indicator for user
      - \* Privacy indicator for public
      - \* Watching indicator
      - \* Being observed indicator
    - \* Ambient Interference
      - \* Visual
      - \* Audio
      - \* Tactile
- \* Location.
  - \* Place\_name

- \* Latitude
- \* Longitude
- \* Altitude
- \* Room
- \* Floor
- \* Building
- \* Address
- \* Street
- \* City
- \* County
- \* State
- \* Country
- \* Postal\_Code
- \* Physiology.
  - \* Pulse
  - \* Body\_temperature
  - \* Blood\_pressure
  - \* Respiration
- \* Activity
  - \* Driving
  - \* Eating
  - \* Running
  - \* Sleeping
  - \* Talking
  - \* Typing
  - \* Walking
- \*Cognitive state

- \* Meaning
- \* Cognition
  - \* Divided User Attention
  - \* Task Switching
  - \* Background Awareness
- \* Solitude
- \* Privacy
  - \* Desired Privacy
  - \* Perceived Privacy
- \* Social Context
- \* Affect
- \* Social state
  - \* Whether the user is alone or if others are present
  - \* Whether the user is being observed (e.g., by a camera)
  - \* The user's perceptions of the people around them and the user's perceptions of the intentions of the people that surround them.
  - \* The user's social role (e.g. they are a prisoner, they are a guard, they are a nurse, they are a teacher, they are a student, etc.)

#### COGNITIVE AVAILABILITY

[00512] There are three kinds of user tasks: focus, routine, and awareness and three main categories of user attention: background awareness, task switched attention, and parallel. Each type of task is associated with a different category of attention. Focus tasks require the highest amount of user attention and are typically associated with task-switched attention. Routine tasks require a minimal amount of user attention or a user's divided attention and are typically associated with parallel attention. Awareness tasks appeals to a user's precognitive state or attention and are typically associated with background awareness. When there is

an abrupt change in the sound, such as changing from a trickle to a waterfall, the user is notified of the change in activity.

## BACKGROUND AWARENESS

- [00513] Background awareness is a non-focus output stimulus that allows the user to monitor information without devoting significant attention or cognition.

### EXAMPLE BACKGROUND AWARENESS

#### CHARACTERIZATION VALUES

- [00514] This characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: the user has no awareness of the computing system/the user has background awareness of the computing system.

- [00515] Using these values as scale endpoints, the following list is an example background awareness scale.

- \* No background awareness is available. A user's pre-cognitive state is unavailable.

- \* A user has enough background awareness available to the computing system to receive one type of feedback or status.

- \* A user has enough background awareness available to the computing system to receive more than one type of feedback, status and so on.

- \* A user's background awareness is fully available to the computing system. A user has enough background awareness available for the computing system such that they can perceive more than two types of feedback or status from the computing system.

### EXEMPLARY UI DESIGN IMPLEMENTATION

#### FOR BACKGROUND AWARENESS

- [00516] The following list contains examples of UI design implementations for how a computing system might respond to a change in background awareness.



- \* If a user does not have any attention for the computing system, that implies that no input or output are needed.

- \* If a user has enough background awareness available to receive one type of feedback, the UI might:

- \* Present a single light in the peripheral vision of a user. For example, this light can represent the amount of battery power available to the computing system. As the battery life weakens, the light gets dimmer. If the battery is recharging, the light gets stronger.

- \* If a user has enough background awareness available to receive more than one type of feedback, the UI might:

- \* Present a single light in the peripheral vision of the user that signifies available battery power and the sound of water to represent data connectivity.

- \* If a user has full background awareness, then the UI might:

- \* Present a single light in the peripheral vision of the user that signifies available battery power, the sound of water that represents data connectivity, and pressure on the skin to represent the amount of memory available to the computing system.

## TASK SWITCHED ATTENTION

[00517] When the user is engaged in more than one focus task, the user's attention can be considered to be task switched.

### EXAMPLE TASK SWITCHED ATTENTION

#### CHARACTERIZATION VALUES

[00518] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: the user does not have any attention for a focus task/the user has full attention for a focus task.

[00519] Using these characteristics as the scale endpoints, the following list is an example of a task switched attention scale.

\* A user does not have any attention for a focus task.

\* A user does not have enough attention to complete a simple focus task.

The time between focus tasks is long.

\* A user has enough attention to complete a simple focus task. The time between focus tasks is long.

\* A user does not have enough attention to complete a simple focus task.

The time between focus tasks is moderately long.

\* A user has enough attention to complete a simple focus task. The time between tasks is moderately long.

\* A user does not have enough attention to complete a simple focus task.

The time between focus tasks is short.

\* A user has enough attention to complete a simple focus task. The time between focus tasks is short.

\* A user does not have enough attention to complete a moderately complex focus task. The time between focus tasks is long.

\* A user has enough attention to complete a moderately complex focus task. The time between focus tasks is long.

\* A user does not have enough attention to complete a moderately complex focus task. The time between focus tasks is moderately long.

\* A user has enough attention to complete a moderately complex focus task. The time between tasks is moderately long.

\* A user does not have enough attention to complete a moderately complex focus task. The time between focus tasks is short.

\* A user has enough attention to complete a moderately complex focus task. The time between focus tasks is short.

\* A user does not have enough attention to complete a moderately complex focus task. The time between focus tasks is long.

\* A user has enough attention to complete a complex focus task. The time between focus tasks is long.

\* A user does not have enough attention to complete a complex focus task. The time between focus tasks is moderately long.

\* A user has enough attention to complete a complex focus task. The time between tasks is moderately long.

\* A user does not have enough attention to complete a complex focus task. The time between focus tasks is short.

\* A user has enough attention to complete a complex focus task. The time between focus tasks is short.

\* A user has enough attention to complete a very complex, multi-stage focus task before moving to a different focus task.

#### PARALLEL

[00520] Parallel attention can consist of focus tasks interspersed with routine tasks (focus task + routine task) or a series of routine tasks (routine task + routine task).

#### EXAMPLE PARALLEL ATTENTION

#### CHARACTERIZATION VALUES

[00521] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: the user does not have enough attention for a parallel task/the user has full attention for a parallel task.

[00522] Using these characteristics as scale endpoints, the following list is an example of a parallel attention scale.

\* A user has enough available attention for one routine task and that task is not with the computing system.

\* A user has enough available attention for one routine task and that task is with the computing system.

\* A user has enough attention to perform two routine tasks and at least of the routine tasks is with the computing system.

\* A user has enough attention to perform a focus task and a routine task. At least one of the tasks is with the computing system.

\* A user has enough attention to perform three or more parallel tasks and at least one of those tasks is in the computing system.

#### PHYSICAL AVAILABILITY

[00523] Physical availability is the degree to which a person is able to perceive and manipulate a device. For example, an airplane mechanic who is repairing an engine does not have hands available to input indications to the computing systems by using a keyboard.

#### LEARNING PROFILE

[00524] A user's learning style is based on their preference for sensory intake of information. That is, most users have a preference for which sense they use to assimilate new information.

#### EXAMPLE LEARNING STYLE CHARACTERIZATION VALUES

[00525] This characterization is enumerated. The following list is an example of learning style characterization values.

- \* Auditory
- \* Visual
- \* Tactile

#### EXEMPLARY UI DESIGN IMPLEMENTATION FOR LEARNING STYLE

[00526] The following list contains examples of UI design implementations for how the computing system might respond to a learning style.

- \* If a user is a auditory learner, the UI might:

- \* Present content to the user by using audio more frequently.
- \* Limit the amount of information presented to a user if there is a lot of ambient noise.
- \* If a user is a visual learner, the UI might:
  - \* Present content to the user in a visual format whenever possible.
  - \* Use different colors to group different concepts or ideas together.
  - \* Use illustrations, graphs, charts, and diagrams to demonstrate content when appropriate.
- \* If a user is a tactile learner, the UI might:
  - \* Present content to the user by using tactile output.
  - \* Increase the affordance of tactile methods of input (e.g. increase the affordance of keyboards).

#### SOFTWARE ACCESSIBILITY

[00527] If an application requires a media-specific plug-in, and the user does not have a network connection, then a user might not be able to accomplish a task.

#### EXAMPLE SOFTWARE ACCESSIBILITY

##### CHARACTERIZATION VALUES

[00528] This characterization is enumerated. The following list is an example of software accessibility values.

- \* The computing system does not have access to software.
- \* The computing system has access to some of the local software resources.
- \* The computing system has access to all of the local software resources.
- \* The computing system has access to all of the local software resources and some of the remote software resources by availing itself to opportunistic user of software resources.

\* The computing system has access to all of the local software resources and all remote software resources by availing itself to the opportunistic user of software resources.

\* The computing system has access to all software resources that are local and remote.

### PERCEPTION OF SOLITUDE

[00529] Solitude is a user's desire for, and perception of, freedom from input. To meet a user's desire for solitude, the UI can do things like:

- \* Cancel unwanted ambient noise
- \* Block out human made symbols generated by other humans and machines

### EXAMPLE SOLITUDE CHARACTERIZATION VALUES

[00530] This characterization is scalar, with the minimum range being binary. Example binary values, or scalar endpoints, are: no solitude/complete solitude.

[00531] Using these characteristics as scale endpoints, the following list is an example of a solitude scale.

- \* No solitude
- \* Some solitude
- \* Complete solitude

### PRIVACY

[00532] Privacy is the quality or state of being apart from company or observation. It includes a user's trust of audience. For example, if a user doesn't want anyone to know that they are interacting with a computing system (such as a wearable computer), the preferred output device might be a head mounted display (HMD) and the preferred input device might be an eye-tracking device.

## HARDWARE AFFINITY FOR PRIVACY

[00533] Some hardware suits private interactions with a computing system more than others. For example, an HMD is a far more private output device than a desk monitor. Similarly, an earphone is more private than a speaker.

[00534] The UI should choose the correct input and output devices that are appropriate for the desired level of privacy for the user's current context and preferences.

## EXAMPLE PRIVACY CHARACTERIZATION VALUES

[00535] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: not private/private, public/not public, and public/private.

[00536] Using no privacy and fully private as the scale endpoints, the following list is an example privacy characterization scale.

- \* No privacy is needed for input or output interaction
- \* The input must be semi-private. The output does not need to be private.
- \* The input must be fully private. The output does not need to be private.
- \* The input must be fully private. The output must be semi-private.
- \* The input does not need to be private. The output must be fully private.
- \* The input does not need to be private. The output must be semi-private.
- \* The input must be semi-private. The output must be semi-private.
- \* The input and output interaction must be fully private.

[00537] \* Semi-private. The user and at least one other person can have access to or knowledge of the interaction between the user and the computing system.

[00538] \* Fully private. Only the user can have access to or knowledge of the interaction between the user and the computing system.

## EXEMPLARY UI DESIGN IMPLEMENTATION FOR PRIVACY

[00539] The following list contains examples of UI design implementations for how the computing system might respond to a change in task complexity.

- \* If no privacy is needed for input or output interaction:

- \* The UI is not restricted to any particular I/O device for presentation and interaction. For example, the UI could present content to the user through speakers on a large monitor in a busy office.

- \* If the input must be semi-private and if the output does not need to be private, the UI might:

- \* Encourage the user to use coded speech commands or use a keyboard if one is available. There are no restrictions on output presentation.

- \* If the input must be fully private and if the output does not need to be private, the UI might:

- \* Not allow speech commands. There are no restrictions on output presentation.

- \* If the input must be fully private and if the output needs to be semi-private, the UI might:

- \* Not allow speech commands (allow only keyboard commands). Not allow an LCD panel and use earphones to provide audio response to the user.

- \* If the output must be semi-private and if the input does not need to be private, the UI might:

- \* Restrict users to an HMD device and/or an earphone for output. There are no restrictions on input interaction.

- \* If the output must be semi-private and if the input does not need to be private, the UI might:



\* Restrict users to HMD devices, earphones, and/or LCD panels. There are no restrictions on input interaction.

\* If the input and output must be semi-private, the UI might:

\* Encourage the user to use coded speech commands and keyboard methods for input. Output may be restricted to HMD devices, earphones or LCD panels.

\* If the input and output interaction must be completely private, the UI might:

\* Not allow speech commands and encourage the user of keyboard methods of input. Output is restricted to HMD devices and/or earphones.

#### USER EXPERTISE

[00540] As the user becomes more familiar with the computing system or the UI, they may be able to navigate through the UI more quickly. Various levels of user expertise can be accommodated. For example, instead of configuring all the settings to make an appointment, users can recite all the appropriate commands in the correct order to make an appointment. Or users might be able to use shortcuts to advance or move back to specific screens in the UI. Additionally, expert users may not need as many prompts as novice users. The UI should adapt to the expertise level of the user.

#### EXAMPLE USER EXPERTISE CHARACTERIZATION VALUES

[00541] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: new user/not new user, not an expert user/expert user, new user/expert user, and novice/expert.

[00542] Using novice and expert as scale endpoints, the following list is an example user expertise scale.

\* The user is new to the computing system and to computing in general.

\* The user is new to the computing system and is an intermediate computer user.

\* The user is new to the computing system, but is an expert computer user.

\* The user is an intermediate user in the computing system.

\* The user is an expert user in the computing system.

#### EXEMPLARY UI DESIGN IMPLEMENTATION FOR USER EXPERTISE

[00543] The following are characteristics of an exemplary audio UI design for novice and expert computer users.

\* The computing system speaks a prompt to the user and waits for a response.

\* If the user responds in x seconds or less, then the user is an expert. The computing system gives the user prompts only.

\* If the user responds in >x seconds, then the user is a novice and the computing system begins enumerating the choices available.

[00544] This type of UI design works well when more than 1 user accesses the same computing system and the computing system and the users do not know if they are a novice or an expert.

#### LANGUAGE

[00545] User context may include language, as in the language they are currently speaking (e.g. English, German, Japanese, Spanish, etc.).

#### EXAMPLE LANGUAGE CHARACTERIZATION VALUES

[00546] This characteristic is enumerated. Example values include:

\* American English

\* British English

\* German

- \* Spanish
- \* Japanese
- \* Chinese
- \* Vietnamese
- \* Russian
- \* French

### COMPUTING SYSTEM

[00547] This section describes attributes associated with the computing system that may cause a UI to change.

[00548] Computing hardware capability

[00549] For purposes of user interfaces designs, there are four categories of hardware:

- \* Input/output devices
- \* Storage (e.g. RAM)
- \* Processing capabilities
- \* Power supply

[00550] The hardware discussed in this topic can be the hardware that is always available to the computing system. This type of hardware is usually local to the user. Or the hardware could sometimes be available to the computing system. When a computing system uses resources that are sometimes available to it, this can be called an opportunistic use of resources.

### STORAGE

[00551] Storage capacity refers to how much random access memory (RAM) is available to the computing system at any given moment. This number is not considered to be constant because the computing system might avail itself to the opportunistic use of memory.

[00552] Usually the user does not need to be aware of how much storage is available unless they are engaged in a task that might require more memory than to which they reliably have access. This might happen when the computing system engages in opportunistic use of memory. For example, if an in-motion user is engaged in a task that requires the opportunistic use of memory and that user decides to change location (e.g. they are moving from their vehicle to a utility pole where they must complete other tasks), the UI might warn the user that if they leave the current location, the computing system may not be able to complete the task or the task might not get completed as quickly.

#### EXAMPLE STORAGE CHARACTERIZATION VALUES

[00553] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no RAM is available/all RAM is available.

[00554] Using no RAM is available and all RAM is available, the following table lists an example storage characterization scale.

Scale attribute	Implication
No RAM is available to the computing system	If no RAM is available, there is no UI available.—Or—There is no change to the UI.
Of the RAM available to the computing system, only the opportunistic use of RAM is available.	The UI is restricted to the opportunistic use of RAM.
Of the RAM that is available to the computing system, only the local RAM is accessible	The UI is restricted to using local RAM.
Of the RAM that is available to	The UI might warn the user that if

Scale attribute	Implication
the computing system, the local RAM is available and the user is about to lose opportunistic use of RAM.	they lose opportunistic use of memory, the computing system might not be able to complete the task, or the task might not be completed as quickly.
Of the total possible RAM available to the computing system, all of it is available.	If there is enough memory available to the computing system to fully function at a high level, the UI may not need to inform the user. If the user indicates to the computing system that they want a task completed that requires more memory, the UI might suggest that the user change locations to take advantage of additional opportunistic use of memory.

## PROCESSING CAPABILITIES

[00567] Processing capabilities fall into two general categories:

\* Speed. The processing speed of a computing system is measured in megahertz (MHz). Processing speed can be reflected as the rate of logic calculation and the rate of content delivery. The more processing power a computing system has, the faster it can calculate logic and deliver content to the user.

\* CPU usage. The degree of CPU usage does not affect the UI explicitly. With current UI design, if the CPU becomes too busy, the UI Typically "freezes" and the user is unable to interact with the computing system. If the CPU usage is too high, the UI will change to accommodate the CPU capabilities. For example,

if the processor cannot handle the demands, the UI can simplify to reduce demand on the processor.

## EXAMPLE PROCESSING CAPABILITY CHARACTERIZATION VALUES

[00568] This UI characterization is scalar, with the minimum range being binary. Example binary or scale endpoints are: no processing capability is available/all processing capability is available.

[00569] Using no processing capability is available and all processing capability as scale endpoints, the following table lists an example processing capability scale.

Scale attribute	Implication
No processing power is available to the computing system	There is no change to the UI.
The computing system has access to a slower speed CPU.	The UI might be audio or text only.
The computing system has access to a high speed CPU	The UI might choose to use video in the presentation instead of a still picture.
The computing system has access to and control of all processing power available to the computing system.	There are no restrictions on the UI based on processing power.

## POWER SUPPLY

[00580] There are two types of power supplies available to computing systems: alternating current (AC) and direct current (DC). Specific scale attributes for AC power supplies are represented by the extremes of the exemplary scale. However, if a user is connected to an AC power supply, it may be useful for the UI to warn an in-motion user when they're leaving an AC power supply. The user might need

to switch to a DC power supply if they wish to continue interacting with the computing system while in motion. However, the switch from AC to DC power should be an automatic function of the computing system and not a function of the UI.

[00581] On the other hand, many computing devices, such as wearable personal computers (WPCs), laptops, and PDAs, operate using a battery to enable the user to be mobile. As the battery power wanes, the UI might suggest the elimination of video presentations to extend battery life. Or the UI could display a VU meter that visually demonstrates the available battery power so the user can implement their preferences accordingly.

#### EXAMPLE POWER SUPPLY

##### CHARACTERIZATION VALUES

[00582] This task characterization is binary if the power supply is AC and scalar if the power supply is DC. Example binary values are: no power/full power. Example scale endpoints are: no power/all power.

[00583] Using no power and full power as scale endpoints, the following list is an example power supply scale.

- \* There is no power to the computing system.
- \* There is an imminent exhaustion of power to the computing system.
- \* There is an inadequate supply of power to the computing system.
- \* There is a limited, but potentially inadequate supply of power to the computing system.
- \* There is a limited but adequate power supply to the computing system.
- \* There is an unlimited supply of power to the computing system.

## EXEMPLARY UI DESIGN IMPLEMENTATION FOR POWER SUPPLY

[00584] The following list contains examples of UI design implementations for how the computing system might respond to a change in the power supply capacity.

- \* If there is minimal power remaining in a battery that is supporting a computing system, the UI might:

- \* Power down any visual presentation surfaces, such as an LCD.

- \* Use audio output only.

- \* If there is minimal power remaining in a battery and the UI is already audio-only, the UI might:

- \* Decrease the audio output volume.

- \* Decrease the number of speakers that receive the audio output or use earplugs only.

- \* Use mono versus stereo output.

- \* Decrease the number of confirmations to the user.

- \* If there is, for example, six hours of maximum-use battery life available and the computing system determines that the user not have access to a different power source for 8 hours, the UI might:

- \* Decrease the luminosity of any visual display by displaying line drawings instead of 3-dimensional illustrations.

- \* Change the chrominance from color to black and white.

- \* Refresh the visual display less often.

- \* Decrease the number of confirmations to the user.

- \* Use audio output only.

- \* Decrease the audio output volume.



## COMPUTING HARDWARE CHARACTERISTICS

[00585] The following is a list of some of the other hardware characteristics that may be influence what is an optimal UI design.

- \* Cost
- \* Waterproof
- \* Ruggedness
- \* Mobility

[00586] Again, there are other characteristics that could be added to this list. However, it is not possible to list all computing hardware attributes that might influence what is considered to be an optimal UI design until run time.

## BANDWIDTH

[00587] There are different types of bandwidth, for instance:

- \* Network bandwidth
- \* Inter-device bandwidth

## NETWORK BANDWIDTH

[00588] Network bandwidth is the computing system's ability to connect to other computing resources such as servers, computers, printers, and so on. A network can be a local area network (LAN), wide area network (WAN), peer-to-peer, and so on. For example, if the user's preferences are stored at a remote location and the computing system determines that the remote resources will not always be available, the system might cache the user's preferences locally to keep the UI consistent. As the cache may consume some of the available RAM resources, the UI might be restricted to simpler presentations, such as text or audio only.

[00589] If user preferences cannot be cached, then the UI might offer the user choices about what UI design families are available and the user can indicate their design family preference to the computing system.

## EXAMPLE NETWORK BANDWIDTH CHARACTERIZATION VALUES

[00590] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no network access/full network access.

[00591] Using no network access and full network access as scale endpoints, the following table lists an example network bandwidth scale.

Scale attribute	Implication
The computing system does not have a connection to network resources.	The UI is restricted to using local computing resources only. If user preferences are stored remotely, then the UI might not account for user preferences.
The computing system has an unstable connection to network resources.	The UI might warn the user that the connection to remote resources might be interrupted. The UI might ask the user if they want to cache appropriate information to accommodate for the unstable connection to network resources.
The computing system has a slow connection to network resources.	The UI might simplify, such as offer audio or text only, to accommodate for the slow connection. Or the computing system might cache appropriate data for the UI so the UI can always be optimized without restriction of the slow connection.

Scale attribute	Implication
The computing system has a high speed, yet limited (by time) access to network resources.	In the present moment, the UI does not have any restrictions based on access to network resources. If the computing system determines that it will lose a network connection, then the UI can warn the user and offer choices, such as does the user want to cache appropriate information, about what to do.
The computing system has a very high-speed connection to network resources.	There are no restrictions to the UI based on access to network resources. The UI can offer text, audio, video, haptic output, and so on.

#### INTER-DEVICE BANDWIDTH

[00604] Inter-device bandwidth is the ability of the devices that are local to the user to communicate with each other. Inter-device bandwidth can affect the UI in that if there is low inter-device bandwidth, then the computing system cannot compute logic and deliver content as quickly. Therefore, the UI design might be restricted to a simpler interaction and presentation, such as audio or text only. If bandwidth is optimal, then there are no restrictions on the UI based on bandwidth. For example, the UI might offer text, audio, and 3-D moving graphics if appropriate for the user's context.

## EXAMPLE INTER-DEVICE BANDWIDTH

### CHARACTERIZATION VALUES

[00605] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no inter-device bandwidth/full inter-device bandwidth.

[00606] Using no inter-device bandwidth and full inter-device bandwidth as scale endpoints, the following table lists an example inter-device bandwidth scale.

Scale attribute	Implication
The computing system does not have inter-device connectivity.	Input and output is restricted to each of the disconnected devices. The UI is restricted to the capability of each device as a stand-alone device.
Some devices have connectivity and others do not.	It depends
The computing system has slow inter-device bandwidth.	The task that the user wants to complete might require more bandwidth that is available among devices. In this case, the UI can offer the user a choice. Does the user want to continue and encounter slow performance? Or, does the user want to acquire more bandwidth by moving to a different location and taking advantage of opportunistic use of bandwidth?
The computing system has fast inter-device bandwidth.	There are few, if any, restrictions on the interaction and presentation between the user and the computing

Scale attribute	Implication
	system. The UI sends a warning message only if there is not enough bandwidth between devices.
The computing system has very high-speed inter-device connectivity.	There are no restrictions on the UI based on inter-device connectivity.

### CONTEXT AVAILABILITY

[00619] Context availability is related to whether the information about the model of the user context is accessible. If the information about the model of the context is intermittent, deemed inaccurate, and so on, then the computing system might not have access to the user's context.

### EXAMPLE CONTEXT AVAILABILITY

#### CHARACTERIZATION VALUES

[00620] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: context not available/context available.

[00621] Using context not available and context available as scale endpoints, the following list is an example context availability scale.

- \* No context is available to the computing system
- \* Some of the user's context is available to the computing system.
- \* A moderate amount of the user's context is available to the computing system.
- \* Most of the user's context is available to the computing system.
- \* All of the user's context is available to the computing system

## EXEMPLARY UI DESIGN FOR CONTEXT AVAILABILITY

[00622] The following list contains examples of UI design implementations for how the computing system might respond to a change in context availability.

\* If the information about the model of context is intermittent, deemed inaccurate, or otherwise unavailable, the UI might:

- \* Stay the same.
- \* Ask the user if the UI needs to change.
- \* Infer a UI from a previous pattern if the user's context history is available.
- \* Change the UI based on all other attributes except for user context (e.g. I/O device availability, privacy, task characteristics, etc.)
- \* Use a default UI.

## OPPORTUNISTIC USE OF RESOURCES

[00623] Some UI components, or other enabling UI content, may allow acquisition from outside sources. For example, if a person is using a wearable computer and they sit at a desk that has a monitor on it, the wearable computer might be able to use the desktop monitor as an output device.

## EXAMPLE OPPORTUNISTIC USE OF RESOURCES CHARACTERIZATION SCALE

[00624] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: no opportunistic use of resources/use of all opportunistic resources.

[00625] Using these characteristics, the following list is an example of an opportunistic use of resources scale.

- \* The circumstances do not allow for the opportunistic use of resources in the computing system.

\* Of the resources available to the computing system, there is a possibility to make opportunistic use of resources.

\* Of the resources available to the computing system, the computing system can make opportunistic use of most of the resources..

\* Of the resources available to the computing system, all are accessible and available.

[00626] Additional information corresponding to this list can be found below in sections related to exemplary scale for storage, exemplary scale for processing capability, and exemplary scale for power supply

## CONTENT

[00627] Content is defined as information or data that is part of or provided by a task. Content, in contrast to UI elements, does not serve a specific role in the user/computer dialog. It provides informative or entertaining information to the user. It is not a control. For example a radio has controls (knobs, buttons) used to choose and format (tune a station, adjust the volume & tone) of broadcasted audio content.

[00628] Sometimes content has associated metadata, but it is not necessary.

[00629] Example content characterization values

[00630] This characterization is enumerated. Example values include:

- \* Quality
- \* Static/streamlined
- \* Passive/interactive
- \* Type
- \* Output device required
- \* Output device affinity
- \* Output device preference
- \* Rendering software

\* Implicit. The computing system can use characteristics that can be inferred from the information itself, such as message characteristics for received messages.

\* Source. A type or instance of carrier, media, channel or network path

\* Destination. Address used to reach the user (e.g., a user typically has multiple address, phone numbers, etc.)

\* Message content. (parseable or described in metadata)

\* Data format type.

\* Arrival time.

\* Size.

\* Previous messages. Inference based on examination of log of actions on similar messages.

\* Explicit. Many message formats explicitly include message-characterizing information, which can provide additional filtering criteria.

\* Title.

\* Originator identification. (e.g., email author)

\* Origination date & time

\* Routing. (e.g., email often shows path through network routers)

\* Priority

\* Sensitivity. Security levels and permissions

\* Encryption type

\* File format. Might be indicated by file name extension

\* Language. May include preferred or required font or font type

\* Other recipients (e.g., email cc field)

\* Required software

\* Certification. A trusted indication that the offer characteristics are dependable and accurate.



\* Recommendations. Outside agencies can offer opinions on what information may be appropriate to a particular type of user or situation.

## SECURITY

[00631] Controlling security is controlling a user's access to resources and data available in a computing system. For example when a user logs on a network, they must supply a valid user name and password to gain access to resource on the network such as, applications, data, and so on.

[00632] In this sense, security is associated with the capability of a user or outside agencies in relation to a user's data or access to data, and does not specify what mechanisms are employed to assure the security.

[00633] Security mechanisms can also be separately and specifically enumerated with characterizing attributes.

[00634] Permission is related to security. Permission is the security authorization presented to outside people or agencies. This characterization could inform UI creation/selection by giving a distinct indication when the user is presented information that they have given permission to others to access.

## EXAMPLE SECURITY CHARACTERIZATION

### VALUES

[00635] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints are: no authorized user access/all user access, no authorized user access/public access, and no public access/public access.

[00636] Using no authorized user access and public access as scale endpoints, the following list is an example security scale.

- \* No authorized access.
- \* Single authorized user access.
- \* Authorized access to more than one person
- \* Authorized access for more than one group of people

\* Public access

[00637] \* Single authorized user only access. The only person who has authorized access to the computing system is a specific user with valid user credentials.

[00638] \* Public access. There are no restrictions on who has access to the computing system. Anyone and everyone can access the computing system.

### TASK CHARACTERIZATIONS

[00639] A task is a user-perceived objective comprising steps. The topics in this section enumerate some of the important characteristics that can be used to describe tasks. In general, characterizations are needed only if they require a change in the UI design.

[00640] The topics in this section include examples of task characterizations, example characterization values, and in some cases, example UI designs or design characteristics.

### TASK LENGTH

[00641] Whether a task is short or long depends upon how long it takes a target user to complete the task. That is, a short task takes a lesser amount of time to complete than a long task. For example, a short task might be creating an appointment. A long task might be playing a game of chess.

### EXAMPLE TASK LENGTH CHARACTERIZATION VALUES

[00642] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: short/not short, long/not long, or short/long.

[00643] Using short/long as scale endpoints, the list is an example task length scale.

\* The task is very short and can be completed in 30 seconds or less

\* The task is moderately short and can be completed in 31-60 seconds.

- \* The task is short and can be completed in 61-90 seconds.
- \* The task is slightly long and can be completed in 91-300 seconds.
- \* The task is moderately long and can be completed in 301-1,200 seconds.
- \* The task is long and can be completed in 1,201-3,600 seconds.
- \* The task is very long and can be completed in 3,601 seconds or more.

#### TASK COMPLEXITY

[00644] Task complexity is measured using the following criteria:

- \* Number of elements in the task. The greater the number of elements, the more likely the task is complex.

- \* Element interrelation. If the elements have a high degree of interrelation, then the more likely the task is complex.

- \* User knowledge of structure. If the structure, or relationships, between the elements in the task is unclear, then the more likely the task is considered to be complex.

[00645] If a task has a large number of highly interrelated elements and the relationship between the elements is not known to the user, then the task is considered to be complex. On the other hand, if there are a few elements in the task and their relationship is easily understood by the user, then the task is considered to be well-structured. Sometimes a well-structured task can also be considered simple.

#### EXAMPLE TASK COMPLEXITY CHARACTERIZATION VALUES

[00646] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: simple/not simple, complex/not complex, simple/complex, well-structured/not well-structured, or well-structured/complex.

[00647] Using simple/complex as scale endpoints, the list is an example task complexity scale.

- \* There is one, very simple task composed of 1-5 interrelated elements whose relationship is well understood.

- \* There is one simple task composed of 6-10 interrelated elements whose relationship is understood.

- \* There is more than one very simple task and each task is composed of 1-5 elements whose relationship is well understood.

- \* There is one moderately simple task composed of 11-15 interrelated elements whose relationship is 80-90% understood by the user.

- \* There is more than one simple task and each task is composed of 6-10 interrelated whose relationship is understood by the user.

- \* There is one somewhat simple task composed of 16-20 interrelated elements whose relationship is understood by the user.

- \* There is more than one moderately simple task and each task is composed of 11-15 interrelated elements whose relationship is 80-90% understood by the user.

- \* There is one complex task composed of 21-35 interrelated elements whose relationship is 60-79% understood by the user.

- \* There is more than one somewhat complex task and each task is composed of 16-20 interrelated elements whose relationship is understood by the user.

- \* There is one moderately complex task composed of 36-50 elements whose relationship is 80-90% understood by the user.

- \* There is more than one complex task and each task is composed of 21-35 elements whose relationship is 60-79% understood by the user.

\* There is one very complex task composed of 51 or more elements whose relationship is 40-60% understood by the user.

\* There is more than one complex task and each task is composed of 36-50 elements whose relationship is 40-60% understood by the user.

\* There is more than one very complex task and each part is composed of 51 or more elements whose relationship is 20-40% understood by the user.

#### EXEMPLARY UI DESIGN IMPLEMENTATION FOR TASK COMPLEXITY

[00648] The following list contains examples of UI design implementations for how the computing system might respond to a change in task complexity.

- \* For a task that is long and simple (well-structured), the UI might:
- \* Give prominence to information that could be used to complete the task.
- \* Vary the text-to-speech output to keep the user's interest or attention.
- \* For a task that is short and simple, the UI might:
- \* Optimize to receive input from the best device. That is, allow only input that is most convenient for the user to use at that particular moment.
- \* If a visual presentation is used, such as an LCD panel or monitor, prominence may be implemented using visual presentation only.
- \* For a task that is long and complex, the UI might:
- \* Increase the orientation to information and devices
- \* Increase affordance to pause in the middle of a task. That is, make it easy for a user to stop in the middle of the task and then return to the task.
- \* For a task that is short and complex, the UI might:
- \* Default to expert mode.
- \* Suppress elements not involved in choices directly related to the current task.
- \* Change modality

## TASK FAMILIARITY

[00649] Task familiarity is related to how well acquainted a user is with a particular task. If a user has never completed a specific task, they might benefit from more instruction from the computing environment than a user who completes the same task daily. For example, the first time a car rental associate rents a car to a consumer, the task is very unfamiliar. However, after about a month, the car rental associate is very familiar with renting cars to consumers.

### EXAMPLE TASK FAMILIARITY CHARACTERIZATION VALUES

[00650] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: familiar/not familiar, not unfamiliar/unfamiliar, and unfamiliar/familiar.

[00651] Using unfamiliar and familiar as scale endpoints, the list is an example task familiarity scale.

\* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 1.

\* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 2.

\* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 3.

\* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 4.

\* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 5.

## EXEMPLARY UI DESIGN IMPLEMENTATION FOR TASK FAMILIARITY

[00652] The following list contains examples of UI design implementations for how the computing system might respond to a change in task familiarity.

- \* For a task that is unfamiliar, the UI might:
  - \* Increase task orientation to provide a high level schema for the task.
  - \* Offer detailed help.
  - \* Present the task in a greater number of steps.
  - \* Offer more detailed prompts.
  - \* Provide information in as many modalities as possible.
- \* For a task that is familiar, the UI might:
  - \* Decrease the affordances for help
  - \* Offer summary help
  - \* Offer terse prompts
  - \* Decrease the amount of detail given to the user
  - \* Use auto-prompt and auto-complete (that is, make suggestions based on past choices made by the user).
  - \* The ability to barge ahead is available.
  - \* Use user-preferred modalities.

## TASK SEQUENCE

[00653] A task can have steps that must be performed in a specific order. For example, if a user wants to place a phone call, the user must dial or send a phone number before they are connected to and can talk with another person. On the other hand, a task, such as searching the Internet for a specific topic, can have steps that do not have to be performed in a specific order.

## EXAMPLE TASK SEQUENCE CHARACTERIZATION VALUES

[00654] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: scripted/not scripted, nondeterministic/not nondeterministic, or scripted/nondeterministic.

[00655] Using scripted/nondeterministic as scale endpoints, the following list is an example task sequence scale.

- \* The each step in the task is completely scripted.
- \* The general order of the task is scripted. Some of the intermediary steps can be performed out of order.
- \* The first and last steps of the task are scripted. The remaining steps can be performed in any order.
- \* The steps in the task do not have to be performed in any order.

## EXEMPLARY UI DESIGN IMPLEMENTATION FOR TASK SEQUENCE

[00656] The following list contains examples of UI design implementations for how the computing system might respond to a change in task sequence.

- \* For a task that is scripted, the UI might:
  - \* Present only valid choices.
  - \* Present more information about a choice so a user can understand the choice thoroughly.
  - \* Decrease the prominence or affordance of navigational controls.
- \* For a task that is nondeterministic, the UI might:
  - \* Present a wider range of choices to the user.
  - \* Present information about the choices only upon request by the user.
  - \* Increase the prominence or affordance of navigational controls



## TASK INDEPENDENCE

- [00657] The UI can coach a user through a task or the user can complete the task without any assistance from the UI. For example, if a user is performing a safety check of an aircraft, the UI can coach the user about what questions to ask, what items to inspect, and so on. On the other hand, if the user is creating an appointment or driving home, they might not need input from the computing system about how to successfully achieve their objective.

### EXAMPLE TASK INDEPENDENCE CHARACTERIZATION VALUES

- [00658] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: coached/not coached, not independently executed/independently executed, or coached/independently executed.

- [00659] Using coached/independently executed as scale endpoints, the following list is an example task guidance scale.

- \* Each step in the task is completely scripted.
- \* The general order of the task is scripted. Some of the intermediary steps can be performed out of order.
- \* The first and last steps of the task are scripted. The remaining steps can be performed in any order.
- \* The steps in the task do not have to be performed in any order.

## TASK CREATIVITY

- [00660] A formulaic task is a task in which the computing system can precisely instruct the user about how to perform the task. A creative task is a task in which the computing system can provide general instructions to the user, but the user uses their knowledge, experience, and/or creativity to complete the task. For example, the computing system can instruct the user about how to write a sonnet.

However, the user must ultimately decide if the combination of words is meaningful or poetic.

#### EXAMPLE TASK CREATIVITY CHARACTERIZATION VALUES

[00661] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints could be defined as formulaic/not formulaic, creative/not creative, or formulaic/creative.

[00662] Using formulaic and creative as scale endpoints, the following list is an example task creativity scale.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 1.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 2.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 3.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 4.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 5.

#### SOFTWARE REQUIREMENTS

[00663] Tasks can be intimately related to software requirements. For example, a user cannot create a complicated database without software.

#### EXAMPLE SOFTWARE REQUIREMENTS CHARACTERIZATION VALUES

[00664] This task characterization is enumerated. Example values include:

- \* JPEG viewer

- \* PDF reader

- \* Microsoft Word
- \* Microsoft Access
- \* Microsoft Office
- \* Lotus Notes
- \* Windows NT 4.0
- \* Mac OS 10

### TASK PRIVACY

[00665] Task privacy is related to the quality or state of being apart from company or observation. Some tasks have a higher level of desired privacy than others. For example, calling a physician to receive medical test results has a higher level of privacy than making an appointment for a meeting with a co-worker.

### EXAMPLE TASK PRIVACY CHARACTERIZATION VALUES

[00666] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: private/not private, public/not public , or private/public.

[00667] Using private/public as scale endpoints, the following table is an example task privacy scale.

- \* The task is not public. Anyone can have knowledge of the task.
- \* The task is semi-private. The user and at least one other person have knowledge of the task.
- \* The task is fully private. Only the user can have knowledge of the task.

### HARDWARE REQUIREMENTS

[00668] A task can have different hardware requirements. For example, talking on the phone requires audio input and output while entering information into a database has an affinity for a visual display surface and a keyboard.

## EXAMPLE HARDWARE REQUIREMENTS

### CHARACTERIZATION VALUES

[00669] This task characterization is enumerated. Example values include:

- \* 10 MB available of storage
- \* 1 hour of power supply
- \* A free USB connection

### TASK COLLABORATION

[00670] A task can be associated with a single user or more than one user. Most current computer-assisted tasks are designed as single-user tasks. Examples of collaborative computer-assisted tasks include participating in a multi-player video game or making a phone call.

### EXAMPLE TASK COLLABORATION CHARACTERIZATION VALUES

[00671] This task characterization is binary. Example binary values are single user/collaboration.

### TASK RELATION

[00672] A task can be associated with other tasks, people, applications, and so on. Or a task can stand alone on it's own.

### EXAMPLE TASK RELATION CHARACTERIZATION VALUES

[00673] This task characterization is binary. Example binary values are unrelated task/related task.

### TASK COMPLETION

[00674] There are some tasks that must be completed once they are started and others that do not have to be completed. For example, if a user is scuba diving and is using a computing system while completing the task of decompressing, it is essential that the task complete once it is started. To ensure the physical safety of

the user, the software must maintain continuous monitoring of the user's elapsed time, water pressure, and air supply pressure/quantity. The computing system instructs the user about when and how to safely decompress. If this task is stopped for any reason, the physical safety of the user could be compromised.

#### EXAMPLE TASK COMPLETION CHARACTERIZATION VALUES

[00675] This task characterization is enumerated. Example values are:

- \* Must be completed
- \* Does not have to be completed
- \* Can be paused
- \* Not known

#### TASK PRIORITY

[00676] Task priority is concerned with order. The order may refer to the order in which the steps in the task must be completed or order may refer to the order in which a series of tasks must be performed. This task characteristic is scalar. Tasks can be characterized with a priority scheme, such as (beginning at low priority) entertainment, convenience, economic/personal commitment, personal safety, personal safety and the safety of others. Task priority can be defined as giving one task preferential treatment over another. Task priority is relative to the user. For example, "all calls from mom" may be a high priority for one user, but not another user.

#### EXAMPLE TASK PRIVACY CHARACTERIZATION VALUES

[00677] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are no priority/high priority.

[00678] Using no priority and high priority as scale endpoints, the following list is an example task priority scale.

- \* The current task is not a priority. This task can be completed at any time.

\* The current task is a low priority. This task can wait to be completed until the highest priority, high priority, and moderately high priority tasks are completed.

\* The current task is moderately high priority. This task can wait to be completed until the highest priority and high priority tasks are addressed.

\* The current task is high priority. This task must be completed immediately after the highest priority task is addressed.

\* The current task is of the highest priority to the user. This task must be completed first.

#### TASK IMPORTANCE

[00679] Task importance is the relative worth of a task to the user, other tasks, applications, and so on. Task importance is intrinsically associated with consequences. For example, a task has higher importance if very good or very bad consequences arise if the task is not addressed. If few consequences are associated with the task, then the task is of lower importance.

#### EXAMPLE TASK IMPORTANCE CHARACTERIZATION VALUES

[00680] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are not important/very important.

[00681] Using not important and very important as scale endpoints, the following list is an example task importance scale.

\* The task is not important to the user. This task has an importance rating of "1."

\* The task is of slight importance to the user. This task has an importance rating of "2."

\* The task is of moderate importance to the user. This task has an importance rating of "3."

\* The task is of high importance to the user. This task has an importance rating of "4."

\* The task is of the highest importance to the user. This task has an importance rating of "5."

### TASK URGENCY

[00682] Task urgency is related to how immediately a task should be addressed or completed. In other words, the task is time dependent. The sooner the task should be completed, the more urgent it is.

### EXAMPLE TASK URGENCY CHARACTERIZATION VALUES

[00683] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are not urgent/very urgency.

[00684] Using not urgent and very urgent as scale endpoints, the following list is an example task urgency scale.

\* A task is not urgent. The urgency rating for this task is "1."

\* A task is slightly urgent. The urgency rating for this task is "2."

\* A task is moderately urgent. The urgency rating for this task is "3."

\* A task is urgent. The urgency rating for this task is "4."

\* A task is of the highest urgency and requires the user's immediate attention. The urgency rating for this task is "5."

### EXEMPLARY UI DESIGN IMPLEMENTATION FOR TASK URGENCY

[00685] The following list contains examples of UI design implementations for how the computing system might respond to a change in task urgency.

\* If the task is not very urgent (e.g. a task urgency rating of 1, using the scale from the previous list), the UI might not indicate task urgency.

\* If the task is slightly urgent (e.g. a task urgency rating of 2, using the scale from the previous list), and if the user is using a head mounted display (HMD), the UI might blink a small light in the peripheral vision of the user.

\* If the task is moderately urgent (e.g. a task urgency rating of 3, using the scale from the previous list), and if the user is using an HMD, the UI might make the light that is blinking in the peripheral vision of the user blink at a faster rate.

\* If the task is urgent, (e.g. a task urgency rating of 4, using the scale from the previous list), and if the user is wearing an HMD, two small lights might blink at a very fast rate in the peripheral vision of the user.

\* If the task is very urgent, (e.g. a task urgency rating of 5, using the scale from the previous list), and if the user is wearing an HMD, three small lights might blink at a very fast rate in the peripheral vision of the user. In addition, a notification is sent to the user's direct line of sight that warns the user about the urgency of the task. An audio notification is also presented to the user.

#### TASK CONCURRENCY

[00686] Mutually exclusive tasks are tasks that cannot be completed at the same time while concurrent tasks can be completed at the same time. For example, a user cannot interactively create a spreadsheet and a word processing document at the same time. These two tasks are mutually exclusive. However, a user can talk on the phone and create a spreadsheet at the same time.

#### EXAMPLE TASK CONCURRENCY

#### CHARACTERIZATION VALUES

[00687] This task characterization is binary. Example binary values are mutually exclusive and concurrent.

#### TASK CONTINUITY

[00688] Some tasks can have their continuity or uniformity broken without comprising the integrity of the task, while other cannot be interrupted without



compromising the outcome of the task. The degree to which a task is associated with saving or preserving human life is often associated with the degree to which it can be interrupted. For example, if a physician is performing heart surgery, their task of performing heart surgery is less interruptible than the task of making an appointment.

#### EXAMPLE TASK CONTINUITY CHARACTERIZATION VALUES

[00689] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are interruptible/not interruptible or abort/pause.

[00690] Using interruptible/not interruptible as scale endpoints, the following list is an example task continuity scale.

- \* The task cannot be interrupted.
- \* The task can be interrupted for 5 seconds at a time or less.
- \* The task can be interrupted for 6-15 seconds at a time.
- \* The task can be interrupted for 16-30 seconds at a time.
- \* The task can be interrupted for 31-60 seconds at a time.
- \* The task can be interrupted for 61-90 seconds at a time.
- \* The task can be interrupted for 91-300 seconds at a time.
- \* The task can be interrupted for 301-1,200 seconds at a time.
- \* The task can be interrupted 1,201-3,600 seconds at a time.
- \* The task can be interrupted for 3,601 seconds or more at a time.
- \* The task can be interrupted for any length of time and for any frequency.

#### COGNITIVE LOAD

[00691] Cognitive load is the degree to which working memory is engaged in processing information. The more working memory is used, the higher the

cognitive load. Cognitive load encompasses the following two facets: cognitive demand and cognitive availability.

[00692] Cognitive demand is the number of elements that a user processes simultaneously. To measure the user's cognitive load, the system can combine the following three metrics: number of elements, element interaction, and structure. Cognitive demand is increased by the number of elements intrinsic to the task. The higher the number of elements, the more likely the task is cognitively demanding. Second, cognitive demand is measured by the level of interrelation between the elements in the task. The higher the inter-relation between the elements, the more likely the task is cognitively demanding. Finally, cognitive load is measured by how well revealed the relationship between the elements is. If the structure of the elements is known to the user or if it's easily understood, then the cognitive demand of the task is reduced.

[00693] Cognitive availability is how much attention the user engages in during the computer-assisted task. Cognitive availability is composed of the following:

- \* Expertise. This includes schema and whether or not it is in long term memory
- \* The ability to extend short term memory.
- \* Distraction. A non-task cognitive demand.

#### HOW COGNITIVE LOAD RELATES TO OTHER ATTRIBUTES

[00694] Cognitive load relates to at least the following attributes:

- \* Learner expertise (novice/expert). Compared to novices, experts have an extensive schemata of a particular set of elements and have automaticity, the ability to automatically understand a class of elements while devoting little to no cognition to the classification. For example, a novice reader must examine every letter of the word that they're trying to read. On the other hand, an expert reader

has built a schema so that elements can be “chunked” into groups and accessed as a group rather than a single element. That is, an expert reader can consume paragraphs of text at a time instead of examining each letter.

\* Task familiarity (unfamiliar/familiar). When a novice and an expert come across an unfamiliar task, each will handle it differently. An expert is likely to complete the task either more quickly or successfully because they access schemas that they already have and use those to solve the problem/understand the information. A novice may spend a lot of time developing a new schema to understand the information/solve the problem.

\* Task complexity (simple/complex or well-structured/complex). A complex task is a task whose structure is not well-known. There are many elements in the task and the elements are highly interrelated. The opposite of a complex task is well-structured. An expert is well-equipped to deal with complex problems because they have developed habits and structures that can help them decompose and solve the problem.

\* Task length (short/long). This relates to much a user has to retain in working memory.

\* Task creativity. (formulaic/creative) How well known is the structure of the interrelation between the elements?

#### EXAMPLE COGNITIVE DEMAND

##### CHARACTERIZATION VALUES

[00695] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are cognitively undemanding/cognitively demanding.

## EXEMPLARY UI DESIGN IMPLEMENTATION FOR COGNITIVE LOAD

[00696] A UI design for cognitive load is influenced by a tasks intrinsic and extrinsic cognitive load. Intrinsic cognitive load is the innate complexity of the task and extrinsic cognitive load is how the information is presented. If the information is presented well (e.g. the schema of the interrelation between the elements is revealed), it reduces the overall cognitive load.

[00697] The following list contains examples of UI design implementations for how the computing system might respond to a change cognitive load.

- \* Present information to the user by using more than one channel. For example, present choices visually to the user, but use audio for prompts.

- \* Use a visual presentation to reveal the relationships between the elements. For example if a family tree is revealed, use colors and shapes to represent male and female members of the tree or shapes and colors can be used to represent different family units.

- \* Reduce the redundancy. For example, if the structure of the elements is revealed visually, do not use audio to explain the same structure to the user.

- \* Keep complementary or associated information together. For example, if creating a dialog box so a user can print, create a button that has the word "Print" on it instead of a dialog box that has a question "Do you want to print?" with a button with the work "OK" on it.

## TASK ALTERABILITY

[00698] Some task can be altered after they are completed while others cannot be changed. For example, if a user moves a file to the Recycle Bin, they can later retrieve the file. Thus, the task of moving the file to the Recycle Bin is alterable. However, if the user deletes the file from the Recycle Bin, they cannot retrieve it at a later time. In this situation, the task is irrevocable.

## EXAMPLE TASK ALTERABILITY CHARACTERIZATION VALUES

[00699] This task characterization is binary, with the minimum range being binary. Example binary values or scale endpoints are alterable/not alterable, irrevocable/revocable, or alterable/irrevocable.

### TASK CONTENT TYPE

[00700] This task characteristic describes the type of content to be used with the task. For example, text, audio, video, still pictures, and so on.

### EXAMPLE CONTENT TYPE CHARACTERISTICS VALUES

[00701] This task characterization is an enumeration. Some example values are:

- \* asp
- \* .jpeg
- \* .avi
- \* .jpg
- \* .bmp
- \* .jsp
- \* .gif
- \* .php
- \* .htm
- \* .txt
- \* .html
- \* .wav
- \* .doc
- \* .xls
- \* .mdb
- \* .vbs
- \* .mpg

[00702] Again, this list is meant to be illustrative, not exhaustive.

#### TASK TYPE

[00703] A task can be performed in many types of situations. For example, a task that is performed in an augmented reality setting might be presented differently to the user than the same task that is executed in a supplemental setting.

#### EXAMPLE TASK TYPE CHARACTERISTICS VALUES

[00704] This task characterization is an enumeration. Example values can include:

- \* Supplemental
- \* Augmentative
- \* Mediated

#### METHODS OF EVALUATING ATTRIBUTES

[00705] This section describes some of the ways in which the UI needs can be passed to the computing system.

#### PREDETERMINED LOGIC

[00706] A human, such as a UI Designer, Software Developer, or outside agency (military, school system, employer, etc.,) can create logic at design time that determines which attributes are passed to the computing system and how they are passed to the computing system. For example, a human could prioritize all of the known attributes. If any of those attributes were present, they would take priority in a very specific order, such as safety, privacy, user preferences and I/O device type.

[00707] Predetermined logic can include, but is not limited to, one or more of the following methods:

- \* Numeric key
- \* XML tags
- \* Programmatic interface
- \* Name/value pairs

## NUMERIC KEY

[00708] UI needs characterizations can be exposed to the system with a numeric value corresponding to values of a predefined data structure.

[00709] For instance, a binary number can have each of the bit positions associated with a specific characteristic. The least significant bit may represent task hardware requirements. Therefore a task characterization of decimal 5 would indicate that minimal processing power is required to complete the task.

## XML TAGS

[00710] UI needs can be exposed to the system with a string of characters conforming to the XML structure.

[00711] For instance, a simple and important task could be represented as:  
<Task Characterization> <Task Complexity= "0" Task Length= "9"> </Task Characterization>

[00712] And a context characterization might be represented by the following:

```
<Context Characterization>
<Theme>Work </Theme>
<Bandwidth>High Speed LAN Network Connection</Bandwidth>
<Field of View> 28° </Field of View>
<Privacy> None </Privacy>
</Context Characterization>
```

[00713] And an I/O device characterization might be represented by the following:

```
<IO Device Characterization>
<Input>Keyboard</Input>
<Input>Mouse</Input>
<Output>Monitor</Output>
<Audio>None</Audio>
</IO Device Characterization>
```

[00714] Note: One significant advantage of this mechanism is that it is easily extensible.

### PROGRAMMING INTERFACE

[00715] A task characterization can be exposed to the system by associating a task characteristic with a specific program call.

[00716] For instance:

GetUrgentTask can return a handle to that communicates that task urgency to the UI.

[00717] Or it could be:

GetHMDDevice can return a handle to the computing system that describes a UI for an HMD.

[00718] Or it could be:

GetSecureContext can return a handle to the computing system that describes a UI a high security user context.

### NAME/VALUE PAIRS

[00719] UI needs can be modeled or represented with multiple attributes that each correspond to specific elements of the task (e.g., complexity, cognitive load or task length), user needs (e.g. privacy, safety, preferences, characteristics) and I/O devices (e.g. device type, redundant controls, audio availability, etc.) and the value of an attribute represents a specific measure of that element. For example, for an attribute that represents the task complexity, a value of "5 " represents a specific measurement of complexity. Or an attribute that represents an output device type, a value of "HMD" represents a specific device. Or an attribute that represents the a user's privacy needs, a value of "5 " represents a specific measurement of privacy.

[00720] Each attribute preferably has the following properties: a name, a value, a timestamp and in some cases (user and task attributes) an uncertainty level. For



example, the name of the complexity attribute may be "task complexity" and its value at a particular time may be 5. Associated with the current value may be a timestamp of 08/01/2001 13:07 PST that indicates when the value was generated, and an uncertainty level of +/- 1 degrees. Or the name of the output device type attribute may be "output device," and its value at a particular time may be "HMD" Associated with the current value may be a timestamp of 08/07/2001 13:07 PST that indicates when the value was generated. Or the name of the privacy attribute may be "User Privacy" and its value at a particular time may be 5. Associated with the current value may be a timestamp of 08/01/2001 13:07 PST that indicates when the value was generated, and an uncertainty level of +/- 1 degrees.

#### USER FEEDBACK

- [00721] Another embodiment is for the computing system to implement user feedback. In this embodiment, the computing system is designed to provide choices to the user and seek feedback about what attribute is most important. This can be implemented when a new attribute becomes available at run time. If the computing system does not recognize the attribute, the user can be queried about how to characterize the attribute. For example, if task privacy had not been previously characterized, the computing system could query the user about how to handle the task (e.g. which I/O devices should be used, hardware affinity, software requirements, and so on).

#### PATTERN RECOGNITION

- [00722] By using pattern recognition algorithms (e.g. neural networks), implicit correlators over time between particular UI designs used and any context attribute (including task, user, and device) can be discovered and used predictively.

## CHARACTERIZING COMPUTER UI DESIGNS WITH RESPECT TO UI REQUIREMENTS

[00723] For a system to accurately choose a UI design that is appropriate or optimal for the user's current computing context, it is useful to determine the design's intended use, required computer configuration, user task, user preferences and other attributes. This section describes an explicit extensible method to characterize UIs.

[00724] In general, any design considerations can be considered when choosing between different UI designs, if they are exposed in a way that the system can interpret.

[00725] This disclosure focuses on the first of the following three types of UI designs:

- \* Supplemental – a software application that runs without integration with the current real world context, such as when the real world context is not even considered.

- \* Augmentative – a software application that presents information in meaningful relationship to the user's perception of the real-world. An example of a UI design characteristic unique to this type of UI design is an indication of whether the design elements are curvaceous or rectilinear. The former is useful when seeking to differentiate the UI elements from man-made environments, the latter from natural environments.

- \* Mediated – a software application that allows the user to perceive and manipulate the real-world from a remote location. An example of a UI design characteristic unique to this type of UI design is whether the design assumes a low time latency between the remote environment and the user (*i.e.*, fast refresh of sounds and images) or one that is optimized for a significant delay.

[00726] There are two important aspects to characterizing UI designs: what UI design attributes are exposed and how are they exposed.

#### CHARACTERIZED ATTRIBUTES

[00727] In some embodiments, a human prepares an explicit characterization of a design before, during, and/or immediately after that UI is designed.

[00728] The characterization can be very simple, such as an indication whether the UI makes use of audio or not. Or the characterization can be arbitrarily complex. For example, one or more of the following attributes could be used to characterize a UI.

- \* Identification (ID). The identifier for a UI design. Any design can have more than one ID. For example, it can have an associated text string designed to be easy to recall by a user, and simultaneously a secure code component that is programmatically recognized.

- \* Source. An identification of the originator or distributor of the design. Like the ID, this can include a user readable description and/or a machine-readable description.

- \* Date. The date for the UI design. Any design can have more than one date. Some relevant dates include when the design was created, updated, or provided.

- \* Version. The version indicates when modifications to existing designs are provided or anticipated.

- \* Input/output device. Many of the methods of presenting or interacting with UI's are dependent on what devices the user can directly manipulate or perceive. Therefore a description of the hardware requirements or affinity is useful.

- \* Cost. Since UI designs can be provided by commercial software vendors, who may or may not require payment, the cost to the consumer may be significant in deciding on whether to use a particular design.

\* Design elements. A UI can be characterized as being composed of particular graphically-described design elements.

\* Functional elements. A UI can be constructed of abstracted UI elements defined by their function, rather than their presentation. A design characterization can include a list of the required elements, allowing the system to choose.

\* Use. A description of intended or appropriate use of a design can be implicit in the characterization of dependencies such as hardware, software, or user profile and preference, or it can be explicitly described. For instance, a design can be characterized as a "deep sea diving" UI.

\* Content. The supported, required, or affinities for specific types of content can be characterized. For instance, a design intended to be used as a virtual radio appliance could enumerate two channels of 44.2 kHz audio as part of its provided content. Or a design could note that though it can display and control motion video, it has been optimized for the slow transition of a series of still images.

[00729] The useful consideration as to whether an attribute should be added to a UI design characterization is whether a change in the attribute would result in the choice of a different design. For example, characterizing the design's intent of working with a head-mounted video display can be important, while noting that the design was created on a Tuesday is not.

#### HOW THE CHARACTERIZATION IS EXPOSED TO THE SYSTEM

[00730] There are many ways to expose the UI's characterization to the system, as shown by the following three examples.

##### NUMERIC KEY

[00731] A UI's characterization can be exposed to the system with a numeric value corresponding to values of a predefined data structure.

[00732] For instance, a binary number can have each of the bit positions associated with a specific characteristic. The least significant bit may represent the need for a visual display device capable of displaying at least 24 characters of text in an unbroken series. Therefore a UI characterization of decimal 5 would require such a display to optimally display its content.

#### XML TAGS

[00733] A UI's characterization can be exposed to the system with a string of characters conforming to the XML structure.

[00734] For instance, a UI design optimized for an audio presentation can include:  
<UI Characterization> <Video Display Required = "0" Audio Output = "1">  
</UI Characterization>

[00735] One significant advantage of the mechanism is that it is easily extensible.

#### PROGRAMMING INTERFACE

[00736] A UI's characterization can be exposed to the system by associating the design with a specific program call.

[00737] For instance:

GetAudioOnlyUI can return a handle to a UI optimized for audio.

#### ILLUSTRATIVE UI DESIGN ATTRIBUTES

[00738] The attributes listed in this spreadsheet are intended to be illustrative. There could be many more attributes that characterize a UI design.

[00739]

Attribute	Description
Content	Characterizes how a UI design presents content to the user. For example, if the UI design is for a LCD, this attribute characterization might communicate to the computing environment that all task content and feedback is on the right side of the display and all user

Attribute	Description
	choices are offered in a menu on the left side of the screen.
Cost	Characterizes the purchase price of the UI design.
Date	The date for the UI design. Any design can have more than one date. Some relevant dates include when the design was created, updated, or provided.
Design elements	Characterizes how the graphically described design elements are assembled in a UI design.
Functional elements	Characterizes how and which abstracted UI elements defined by their function are assembled in a UI design. A design characterization can include a list of the required elements, allowing the system to choose.
Hardware affinity	Characterizes with which hardware the UI design has affinity. This characteristic does not include output devices.
Identification (ID)	The identifier for a UI design. Any design can have more than one ID.
Importance	Characterizes the UI design for task importance.
Input and output devices	Characterizes for which input and output devices have affinity for this particular UI design.
Language	Characterizes for which language(s) the UI design is optimized.
Learning profile	Characterizes the learning style built into the UI.

Attribute	Description
Length	Characterizes how the UI design accommodates the task length.
Name	The name of the UI design.
Physical availability	Characterizes how the UI design accommodates different levels of physical availability (the degree to which user's body or part of their body is in use). For example, a UI designed to work with speech commands accommodates users who hands are physically unavailable because the user is repairing an airplane engine.
Power supply	Characterizes how much power the UI design uses. Typically, this is determined by the type of hardware the design requires.
Priority	Characterizes how the UI design presents task priority.
Privacy	Characterizes the level of privacy built in to the UI design. For example, a UI that is designed to use coded speech commands and a head mounted display is more private than a UI designed to use non-coded speech commands and a desktop monitor.
Processing capabilities	Characterizes the speed and CPU usage required for a UI design.
Safety	Characterizes the safety precautions built into the UI design. For instance, designs that require greater user attention may be characterized as less

Attribute	Description
	safe.
Security	Characterizes a the level of security built into a UI design.
Software capability	Characterizes the ability of the software available to the computing environment.
Source	Indicates the person, organization, business, or otherwise who created the UI design. This attribute can include a user readable description and/or a machine-readable description.
Storage	Characterizes the amount of storage (e.g. RAM) needed by the UI design.
System audio	Characterizes whether the UI is capable to receive audio signals from the user on behalf of the computing environment.
Task complexity	Characterizes the UI design for task complexity. For example, if the UI is output to a visual presentation surface and the task is simple, the entire task might be encapsulated in one screen. If the task is complex, the task might be separated into multiple steps.
Theme	Characterizes a related set of measures of specific context elements, such as ambient temperature and current task, built into the UI design.
Urgency	Characterizes how the UI design presents task urgency to the user.
Use	The explicit characterization of the intended



Attribute	Description
	purpose or use of a UI design. For instance, a design can be characterized as a "deep sea diving" UI.
User attention	Characterizes the UI design for user attention. For example, if the user has full attention for the computing environment, the UI may be more complicated than a UI design for a user who has only background attention for the computing environment.
User audio	Characterizes the UI's ability to present audio signals to the user.
User characteristics	Characterizes how the UI design accommodates for user characteristics such as emotional and physical states.
User expertise	Characterizes how the UI design accommodates user expertise.
User preferences	Characterizes how a UI design accommodates for a set of attributes that reflect user likes and dislikes, such as I/O devices preferences, volume of audio output, amount of haptic pressure, and font size and color for visual display surfaces.
Version	The version indicates when modifications to existing designs are provided or anticipated.
Video	Characterizes whether the UI design presents visual output to the user through a visual presentation surface such as a head mounted display, monitor, or LCD.

## AUTOMATED SELECTION OF APPROPRIATE OR OPTIMAL COMPUTER UI

[00812] This section describes techniques to enable a computing system to change the user interface by choosing from a group of preexisting UI designs at run time. Figure 6 provides an overview of how this is accomplished.

[00813] The left side of figure 6 shows how the characterizations of the user's task functionality, I/O devices local to the user, and context are combined to create a description of the optimal UI for the current situation. The right side of figure 6 shows UI designs that have been explicitly characterized. These optimal UI characterizations are compared to the available UI characterizations and when a match is found, that UI is used.

[00814] To accurately choose which UI design is optimal for the user's current computing context, a system compares a design's intended use to the current requirements for a UI. This disclosure describes an explicit extensible method to dynamically compare the characterizations of UI designs to the characterization of the current UI needs and then choose a UI design based on how the characterizations match run time. Figure 6 shows the overall logic.

### 3001: CHARACTERIZED UI DESIGNS

[00815] Figure 7 illustrates a variety of characterized UI designs 3001. These UI designs can be characterized in various ways, such as by a human preparing an explicit characterization of that design before, during or immediately after a UI is designed. The characterization can be very simple, such as an indication whether the UI makes use of audio or not. Or the characterization can be arbitrarily complex. For example, one or more of the following attributes could be used to characterize a UI.

\* Identification (ID). The identifier for a UI design. Any design can have more than one ID. For example, it can have an associated text string designed to

be easy to recall by a user, and simultaneously a secure code component that is programmatically recognized.

\* Source. An identification of the originator or distributor of the design. Like the ID, this can include a user readable description and/or a machine-readable description.

\* Date. The date for the UI design. Any design can have more than one date. Some relevant dates include when the design was created, updated, or provided.

\* Version. The version indicates when modifications to existing designs are provided or anticipated.

\* Input/output device. Many of the methods of presenting or interacting with UI's are dependent on what devices the user can directly manipulate or perceive. Therefore a description of the hardware requirements or affinity is useful.

\* Cost. Since UI designs can be provided by commercial software vendors, who may or may not require payment, the cost to the consumer may be significant in deciding on whether to use a particular design.

\* Design elements. A UI can be characterized as being composed of particular graphically-described design elements.

\* Functional elements. A UI can be constructed of abstracted UI elements defined by their function, rather than their presentation. A design characterization can include a list of the required elements, allowing the system to choose.

\* Use. A description of intended or appropriate use of a design can be implicit in the characterization of dependencies such as hardware, software, or user profile and preference, or it can be explicitly described. For instance, a design can be characterized as a "deep sea diving" UI.

\* Content. The supported, required, or affinities for specific types of content can be characterized. For instance, a design intended to be used as a virtual

radio appliance could enumerate two channels of 44.2 kHz audio as part of its provided content. Or a design could note that though it can display and control motion video, it has been optimized for the slow transition of a series of still images.

- [00816] The useful consideration as to whether an attribute should be added to a UI design characterization is whether a change in the attribute would result in the choice of a different design. For example, characterizing the design's intent of working with a head-mounted video display can be important, while noting that the design was created on a Tuesday is not.

#### HOW THE CHARACTERIZATION IS EXPOSED TO THE SYSTEM

- [00817] There are many ways to expose the UI's characterization to the system, as shown by the following three examples.

##### NUMERIC KEY

- [00818] A UI's characterization can be exposed to the system with a numeric value corresponding to values of a predefined data structure.

- [00819] For instance, a binary number can have each of the bit positions associated with a specific characteristic. The least significant bit may represent the need for a visual display device capable of displaying at least 24 characters of text in an unbroken series. Therefore a UI characterization of decimal 5 would require such a display to optimally display its content.

##### XML TAGS

- [00820] A UI's characterization can be exposed to the system with a string of characters conforming to the XML structure.

- [00821] For instance, a UI design optimized for an audio presentation can include:  
<UI Characterization> <Video Display Required = "0" Audio Output = "1">  
</UI Characterization>

- [00822] One significant advantage of the mechanism is that it is easily extensible.

## PROGRAMMING INTERFACE

[00823] A UI's characterization can be exposed to the system by associating the design with a specific program call.

[00824] For instance:

GetAudioOnlyUI can return a handle to a UI optimized for audio.

### 3002: OPTIMAL UI CHARACTERIZATIONS

[00825] This section describes modeled real-world and virtual contexts to which the described techniques can respond. The described model for optimal UI design characterization includes at least the following categories of attributes when determining the optimal UI design:

- \* All available attributes. The model is dynamic so it can accommodate for any and all attributes that could affect the optimal UI design for a user's context. For example, this model could accommodate for temperature, weather conditions, time of day, available I/O devices, preferred volume level, desired level of privacy, and so on.

- \* Significant attributes. Some attributes have a more significant influence on the optimal UI design than others. Significant attributes include, but are not limited to, the following:

- \* The user can see video.
- \* The user can hear audio.
- \* The computing system can hear the user.
- \* The interaction between the user and the computing system must be private.
- \* The user's hands are occupied.

- \* Attributes that correspond to a theme. Specific or programmatic.  
Individual or group.

[00826] For clarity, many of the example attributes described in this topic is presented with a scale and some include design examples. It is important to note that any of the attributes mentioned in this document are just examples, however. There are other attributes that can cause a UI to change that are not listed in this document. The described dynamic model can account for additional attributes.

### I/O DEVICES

[00827] Output — Devices that are directly perceivable by the user. For example, a visual output device creates photons that enter the user's eye. Output devices are always local to the user.

[00828] Input — A device that can be directly manipulated by the user. For example, a microphone translates energy created by the user's voice into electrical signals that can control a computer. Input devices are always local to the user.

[00829] The input devices to which the user has access to interact with the computer in ways that convey choices include, but is not limited to:

- \* Keyboards
- \* Touch pads
- \* Mice
- \* Trackballs
- \* Microphones
- \*

Rolling/pointing/pressing/bending/turning/twisting/switching/rubbing/zipping  
cursor controllers - anything that the user's manipulation of can be sensed by the computer, this includes body movement that forms recognizable gestures.

- \* Buttons, etc.

[00830] Output devices allow the presentation of computer-controlled information and content to the user, and includes:

- \* Speakers
- \* Monitors
- \* Pressure actuators, etc.

#### INPUT DEVICE TYPES

[00831] Some characterizations of input devices are a direct result of the device itself.

#### TOUCH SCREEN

[00832] A display screen that is sensitive to the touch of a finger or stylus. Touch screens are very resistant to harsh environments where keyboards might eventually fail. They are often used with custom-designed applications so that the on-screen buttons are large enough to be pressed with the finger. Applications are typically very specialized and greatly simplified so they can be used by anyone. However, touch screens are also very popular on PDAs and full-size computers with standard applications, where a stylus is required for precise interaction with screen objects.

#### EXAMPLE TOUCH SCREEN ATTRIBUTE CHARACTERISTIC VALUES

[00833] This characteristic is enumerated. Some example values are:

- \* Screen objects must be at least 1 centimeter square
- \* The user can see the touch screen directly
- \* The user can see the touch screen indirectly (e.g. by using a monitor)
- \* Audio feedback is available
- \* Spatial input is difficult
- \* Feedback to the user is presented to the user through a visual presentation surface.

#### POINTING DEVICE

[00834] An input device used to move the pointer (cursor) on screen.

## EXAMPLE POINTING DEVICE CHARACTERISTIC VALUES

[00835] This characteristic is enumerated. Some example values are:

- \* 1-dimension (D) pointing device
- \* 2-D pointing device
- \* 3-D pointing device
- \* Position control device
- \* Range control device
- \* Feedback to the user is presented through a visual presentation

surface.

## SPEECH

[00836] The conversion of spoken words into computer text. Speech is first digitized and then matched against a dictionary of coded waveforms. The matches are converted into text as if the words were typed on the keyboard.

## EXAMPLE SPEECH CHARACTERISTIC VALUES

[00837] This characteristic is enumerated. Example values are:

- \* Command and control
- \* Dictation
- \* Constrained grammar
- \* Unconstrained grammar

## KEYBOARD

[00838] A set of input keys. On terminals and personal computers, it includes the standard typewriter keys, several specialized keys and the features outlined below.

## EXAMPLE KEYBOARD CHARACTERISTIC VALUES

[00839] This characteristic is enumerated. Example values are:

- \* Numeric



- \* Alphanumeric
- \* Optimized for discreet input

#### PEN TABLET

[00840] A digitizer tablet that is specialized for handwriting and hand marking. LCD-based tablets emulate the flow of ink as the tip touches the surface and pressure is applied. Non-display tablets display the handwriting on a separate computer screen.

#### EXAMPLE PEN TABLET CHARACTERISTIC VALUES

[00841] This characteristic is enumerated. Example values include:

- \* Direct manipulation device
- \* Feedback is presented to the user through a visual presentation surface
- \* Supplemental feedback can be presented to the user using audio output.
- \* Optimized for special input
- \* Optimized for data entry

#### EYE TRACKING

[00842] An eye-tracking device is a device that uses eye movement to send user indications about choices to the computing system. Eye tracking devices are well suited for situations where there is little to no motion from the user (e.g. the user is sitting at a desk) and has much potential for non-command user interfaces.

#### EXAMPLE EYE TRACKING CHARACTERISTIC VALUES

[00843] This characteristic is enumerated. Example values include:

- \* 2-D pointing device
- \* User motion = still
- \* Privacy = high

## OUTPUT DEVICE TYPES

[00844] Some characterizations of input devices are a direct result of the device itself.

### HMD

[00845] Head Mounted Display) A display system built and worn like goggles that gives the illusion of a floating monitor in front of the user's face. The HMD is an important component of a body-worn computer (wearable computer). Single-eye units are used to display hands-free instructional material, and dual-eye, or stereoscopic, units are used for virtual reality applications.

### EXAMPLE HMD CHARACTERISTIC VALUES

[00846] This characteristic is enumerated. Example values include:

- \* Field of view > 28°
- \* User's hands = not available
- \* User's eyes = forward and out
- \* User's reality = augmented, mediated, or virtual

### MONITORS

[00847] A display screen used to present output from a computer, camera, VCR or other video generator. A monitor's clarity is based on video bandwidth, dot pitch, refresh rate, and convergence.

### EXAMPLE MONITOR CHARACTERISTIC VALUES

[00848] This characteristic is enumerated. Some example values include:

- \* Required graphical resolution = high
- \* User location = stationary
- \* User attention = high
- \* Visual density = high
- \* Animation = yes
- \* Simultaneous presentation of information = yes (e.g. text and image)

- \* Spatial content = yes

#### I/O DEVICE USE

[00849] This attribute characterizes how or for what an input or output device can be optimized for use. For example, a keyboard is optimized for entering alphanumeric text characters and monitor, head mounted display (HMD), or LCD panel is optimized for displaying those characters and other visual information.

#### EXAMPLE DEVICE USE CHARACTERIZATION VALUES

[00850] This characterization is enumerated. Example values include:

- \* Speech recognition
- \* Alphanumeric character input
- \* Handwriting recognition
- \* Visual presentation
- \* Audio presentation
- \* Haptic presentation
- \* Chemical presentation

#### REDUNDANT CONTROLS

[00851] The user may have more than one way to perceive or manipulate the computing environment. For instance, they may be able to indicate choices by manipulating a mouse, or speaking.

[00852] By providing UI designs that have more than one I/O modality (also known as “multi-modal”), greater flexibility can be provided to the user. However, there are times when this is not appropriate. For instance, the devices may not be constantly available (user’s hands are occupied, the ambient noise increases defeating voice recognition).

## EXAMPLE REDUNDANT CONTROLS CHARACTERIZATION VALUES

[00853] As a minimum, a numeric value could be associated with a configuration of devices.

- \* 1 – keyboard and touch screen
- \* 2 – HMD and 2-D pointing device

[00854] Alternately, a standardized list of available, preferred, or historically used devices could be used.

- \* QWERTY keyboard
- \* Twiddler
- \* HMD
- \* VGA monitor
- \* SVGA monitor
- \* LCD display
- \* LCD panel

## PRIVACY

[00855] Privacy is the quality or state of being apart from company or observation. It includes a user's trust of audience. For example, if a user doesn't want anyone to know that they are interacting with a computing system (such as a wearable computer), the preferred output device might be an HMD and the preferred input device might be an eye-tracking device.

## HARDWARE AFFINITY FOR PRIVACY

[00856] Some hardware suits private interactions with a computing system more than others. For example, an HMD is a far more private output device than a desk monitor. Similarly, an earphone is more private than a speaker.

[00857] The UI should choose the correct input and output devices that are appropriate for the desired level of privacy for the user's current context and preferences.

#### EXAMPLE PRIVACY CHARACTERIZATION VALUES

[00858] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: not private/private, public/not public, and public/private.

[00859] Using no privacy and fully private as the scale endpoints, the following table lists an example privacy characterization scale.

Scale attribute	Implication/Example
No privacy is needed for input or output interaction	The UI is not restricted to any particular I/O device for presentation and interaction. For example, the UI could present content to the user through speakers on a large monitor in a busy office.
The input must be semi-private. The output does not need to be private.	Coded speech commands, and keyboard methods are appropriate. No restrictions on output presentation.
The input must be fully private. The output does not need to be private.	No speech commands. No restriction on output presentation.
The input must be fully private. The output must be semi-private.	No speech commands. No LCD panel.
The input does not need to be private. The output must be fully private.	No restrictions on input interaction. The output is restricted to an HMD device and/or an earphone.

Scale attribute	Implication/Example
The input does not need to be private. The output must be semi-private.	No restrictions on input interaction. The output is restricted to HMD device, earphone, and/or an LCD panel.
The input must be semi-private. The output must be semi-private.	Coded speech commands and keyboard methods are appropriate. Output is restricted to an HMD device, earphone or an LCD panel.
The input and output interaction must be fully private.	No speech commands. Keyboard devices might be acceptable. Output is restricted to and HMD device and/or an earphone.

[00878] \* Semi-private. The user and at least one other person can have access to or knowledge of the interaction between the user and the computing system.

[00879] \* Fully private. Only the user can have access to or knowledge of the interaction between the user and the computing system.

#### VISUAL

[00880] Visual output refers to the available visual density of the display surface is characterized by the amount of content a presentation surface can present to a user. For example, an LED output device, desktop monitor, dashboard display, hand-held device, and head mounted display all have different amounts of visual density. UI designs that are appropriate for a desktop monitor are very different than those that are appropriate for head-mounted displays. In short, what is considered to be the optimal UI will change based on what visual output device(s) is available.

[00881] In addition to density, visual display surfaces have the following characteristics:

- \* Color
- \* Motion
- \* Field of view
- \* Depth
- \* Reflectivity
- \* Size. Refers to the actual size of the visual presentation surface.
- \* Position/location of visual display surface in relation to the user and the task that they're performing.
- \* Number of focal points. A UI can have more than one focal point and each focal point can display different information.
- \* Distance of focal points from the user. A focal point can be near the user or it can be far away. The amount distance can help dictate what kind and how much information is presented to the user.
- \* Location of focal points in relation to the user. A focal point can be to the left of the user's vision, to the right, up, or down.
- \* With which eye(s) the output is associated. Output can be associated with a specific eye or both eyes.
- \* Ambient light.
- \* Others

[00882] The topics in this section describe in further detail the characteristics of some of these previously listed attributes.

#### EXAMPLE VISUAL DENSITY CHARACTERIZATION VALUES

[00883] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no visual density/full visual density.

[00884] Using no visual density and full visual density as scale endpoints, the following table lists an example visual density scale.

Scale attribute	Implication/Design example
There is no visual density	The UI is restricted to non-visual output such as audio, haptic, and chemical.
Visual density is very low	The UI is restricted to a very simple output, such as single binary output devices (a single LED) or other simple configurations and arrays of light. No text is possible.
Visual density is low	The UI can handle text, but is restricted to simple prompts or the bouncing ball.
Visual density is medium	The UI can display text, simple prompts or the bouncing ball, and very simple graphics.
Visual density is high	The visual display has fewer restrictions. Visually dense items such as windows, icons, menus, and prompts are available as well as streaming video, detailed graphics and so on.
Visual density is the highest available	The UI is not restricted by visual density. A visual display that mirrors reality (e.g. 3-dimensional) is possible and appropriate.



## COLOR

[00899] This characterizes whether or not the presentation surface displays color. Color can be directly related to the ability of the presentation surface, or it could be assigned as a user preference.

- \* Chrominance. The color information in a video signal.

- \* Luminance. The amount of brightness, measured in lumens, which is given off by a pixel or area on a screen. It is the black/gray/white information in a video signal. Color information is transmitted as luminance (brightness) and chrominance (color). For example, dark red and bright red would have the same chrominance, but a different luminance. Bright red and bright green could have the same luminance, but would always have a different chrominance.

### EXAMPLE COLOR CHARACTERIZATION VALUES

[00900] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no color/full color.

[00901] Using no color and full color as scale endpoints, the following table lists an example color scale.

Scale attribute	Implication/Design example
No color is available.	The UI visual presentation is monochrome.
One color is available.	The UI visual presentation is monochrome plus one color.
Two colors are available	The UI visual presentation is monochrome plus two colors or any combination of the two colors.
Full color is available.	The UI is not restricted by color.

## MOTION

- [00912] This characterizes whether or not a presentation surface has the ability to present motion to the user. Motion can be considered as a stand-alone attribute or as a composite attribute.

### EXAMPLE MOTION CHARACTERIZATION VALUES

- [00913] As a stand-alone attribute, this characterization is binary. Example binary values are: no animation available/animation available.

- [00914] As a composite attribute, this characterization is scalar. Example scale endpoints include no motion/motion available, no animation available/animation available, or no video/video. The values between the endpoints depend on the other characterizations that are included in the composite. For example, the attributes color, visual density, and frames per second, etc. change the values between no motion and motion available.

## FIELD OF VIEW

- [00915] A presentation surface can display content in the focus of a user's vision, in the user's periphery, or both.

### EXAMPLE FIELD OF VIEW CHARACTERIZATION VALUES

- [00916] This UI characterization is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: peripheral vision only/field of focus and peripheral vision is available.
- [00917] Using peripheral vision only and field of focus and peripheral vision is available as scale endpoints, the following tables lists an example field of view scale.

Scale attribute	Implication
All visual display is in the peripheral vision of the user	The UI is restricted to using the peripheral vision of the user. Lights,

Scale attribute	Implication
	colors and other simple visual display are appropriate. Text is not appropriate.
Only the user's field of focus is available.	The UI is restricted to using the users field of vision only. Text and other complex visual displays are appropriate.
Both field of focus and the peripheral vision of the user are used.	The UI is not restricted by the user's field of view.

#### EXEMPLARY UI DESIGN IMPLEMENTATION FOR CHANGES IN FIELD OF VIEW

[00926] The following list contains examples of UI design implementations for how the computing system might respond to a change in field of view.

- \* If the field of view for the visual presentation is more than 28°, then the UI might:

- \* Display the most important information at the center of the visual presentation surface.

- \* Devote more of the UI to text

- \* Use periphicons outside of the field of view.

- \* If the field of view for the visual presentation is less than 28°, then the UI might:

- \* Restrict the size of the font allowed in the visual presentation. For example, instead of listing "Monday, Tuesday, and Wednesday," and so on as choices, the UI might list "M, Tu, W" instead.

- \* The body or environment stabilized image can scroll.

## DEPTH

- [00927] A presentation surface can display content in 2 dimensions (e.g. a desktop monitor) or 3 dimensions (a holographic projection).

### EXAMPLE DEPTH CHARACTERIZATION VALUES

- [00928] This characterization is binary and the values are: 2 dimensions, 3 dimensions.

## REFLECTIVITY

- [00929] The fraction of the total radiant flux incident upon a surface that is reflected and that varies according to the wavelength distribution of the incident radiation.

### EXAMPLE REFLECTIVITY CHARACTERIZATION VALUES

- [00930] This characterization is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: not reflective/highly reflective or no glare/high glare.

- [00931] Using not reflective and highly reflective as scale endpoints, the following list is an example of a reflectivity scale.

- \* Not reflective (no surface reflectivity).
- \* 10% surface reflectivity
- \* 20% surface reflectivity
- \* 30% surface reflectivity
- \* 40% surface reflectivity
- \* 50% surface reflectivity
- \* 60% surface reflectivity
- \* 70% surface reflectivity
- \* 80% surface reflectivity
- \* 90% surface reflectivity
- \* Highly reflective (100% surface reflectivity)

## EXEMPLARY UI DESIGN IMPLEMENTATION FOR CHANGES IN REFLECTIVITY

[00932] The following list contains examples of UI design implementations for how the computing system might respond to a change in reflectivity.

- \* If the output device has high reflectivity — a lot of glare — then the visual presentation will change to a light colored UI.

### AUDIO

[00933] Audio input and output refers to the UI's ability to present and receive audio signals. While the UI might be able to present or receive any audio signal strength, if the audio signal is outside the human hearing range (approximately 20 Hz to 20,000 Hz) it is converted so that it is within the human hearing range, or it is transformed into a different presentation, such as haptic output, to provide feedback, status, and so on to the user

[00934] Factors that influence audio input and output include (but this is not an inclusive list):

- \* Level of ambient noise (this is an environmental characterization)
- \* Directionality of the audio signal
- \* Head-stabilized output (e.g. earphones)
- \* Environment-stabilized output (e.g. speakers)
- \* Spatial layout (3-D audio)
- \* Proximity of the audio signal to the user
- \* Frequency range of the speaker
- \* Fidelity of the speaker, e.g. total harmonic distortion
- \* Left, right, or both ears
- \* What kind of noise is it?
- \* Others

## EXAMPLE AUDIO OUTPUT CHARACTERIZATION VALUES

[00935] This characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: the user cannot hear the computing system/ the user can hear the computing system.

[00936] Using the user cannot hear the computing system and the user can hear the computing system as scale endpoints, the following table lists an example audio output characterization scale.

Scale attribute	Implication
The user cannot hear the computing system.	The UI cannot use audio to give the user choices, feedback, and so on.
The user can hear audible whispers (approximately 10-30 dBA).	The UI might offer the user choices, feedback, and so on by using the earphone only.
The user can hear normal conversation (approximately 50-60 dBA).	The UI might offer the user choices, feedback, and so on by using a speaker(s) connected to the computing system.
The user can hear communications from the computing system without restrictions.	The UI is not restricted by audio signal strength needs or concerns.
Possible ear damage (approximately 85+ dBA)	The UI will not output audio for extended periods of time that will damage the user's hearing.

## EXAMPLE AUDIO INPUT CHARACTERIZATION VALUES

[00949] This characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: the computing system cannot hear the user/the computing system can hear the user.

[00950] Using the computing system cannot hear the user and the computing system can hear the user as scale endpoints, the following table lists an example audio input scale.

Scale attribute	Implication
The computing system cannot receive audio input from the user.	When the computing system cannot receive audio input from the user, the UI will notify the user that audio input is not available.
The computing system is able to receive audible whispers from the user (approximate 10-30 dBA).	
The computing system is able to receive normal conversational tones from the user (approximate 50-60 dBA).	
The computing system can receive audio input from the user without restrictions.	The UI is not restricted by audio signal strength needs or concerns.
The computing system can receive only high volume audio input from the user.	The computing system will not require the user to give indications using a high volume. If a high volume is required, then the UI will notify the user

	that the computing system cannot receive audio input from the user.
--	---

## HAPTICS

[00963] Haptics refers to interacting with the computing system using a tactile method. Haptic input includes the computing system's ability to sense the user's body movement, such as finger or head movement. Haptic output can include applying pressure to the user's skin. For haptic output, the more transducers, the more skin covered, the more resolution for presentation of information. That is if the user is covered with transducers, the computing system receives a lot more input from the user. Additionally, the ability for haptically- oriented output presentations is far more flexible.

### EXAMPLE HAPTIC INPUT CHARACTERIZATION VALUES

[00964] This characteristic is enumerated. Possible values include accuracy, precision, and range of:

- \* Pressure
- \* Velocity
- \* Temperature
- \* Acceleration
- \* Torque
- \* Tension
- \* Distance
- \* Electrical resistance
- \* Texture
- \* Elasticity
- \* Wetness

Additionally, the characteristics listed previously are enhanced by:



\* Number of dimensions

\* Density and quantity of sensors (e.g. a 2 dimensional array of sensors.

The sensors could measure the characteristics previously listed).

#### CHEMICAL OUTPUT

[00965] Chemical output refers to using chemicals to present feedback, status, and so on to the user. Chemical output can include:

\* Things a user can taste

\* Things a user can smell

#### EXAMPLE TASTE CHARACTERISTIC VALUES

[00966] This characteristic is enumerated. Example characteristic values of taste include:

\* Bitter

\* Sweet

\* Salty

\* Sour

#### EXAMPLE SMELL CHARACTERISTIC VALUES

[00967] This characteristic is enumerated. Example characteristic values of smell include:

\* Strong/weak

\* Pungent/bland

\* Pleasant/unpleasant

\* Intrinsic, or signaling

#### ELECTRICAL INPUT

[00968] Electrical input refers to a user's ability to actively control electrical impulses to send indications to the computing system.

\* Brain activity

\* Muscle activity

## EXAMPLE ELECTRICAL INPUT CHARACTERIZATION VALUES

[00969] This characteristic is enumerated. Example values of electrical input can include:

- \* Strength of impulse
- \* Frequency

### USER CHARACTERIZATIONS

[00970] This section describes the characteristics that are related to the user.

### USER PREFERENCES

[00971] User preferences are a set of attributes that reflect the user's likes and dislikes, such as I/O devices preferences, volume of audio output, amount of haptic pressure, and font size and color for visual display surfaces. User preferences can be classified in the following categories:

\* Self characterization. Self-characterized user preferences are indications from the user to the computing system about themselves. The self-characterizations can be explicit or implicit. An explicit, self-characterized user preference results in a tangible change in the interaction and presentation of the UI. An example of an explicit, self-characterized user preference is "Always use the font size 18" or "The volume is always off." An implicit, self-characterized user preference results in a change in the interaction and presentation of the UI, but it might be not be immediately tangible to the user. A learning style is an implicit self-characterization. The user's learning style could affect the UI design, but the change is not as tangible as an explicit, self-characterized user preference. If a user characterizes themselves to a computing system as a "visually impaired, expert computer user," the UI might respond by always using 24-point font and monochrome with any visual display surface. Additionally, tasks would be

chunked differently, shortcuts would be available immediately, and other accommodations would be made to tailor the UI to the expert user.

\* Theme selection. In some situations, it is appropriate for the computing system to change the UI based on a specific theme. For example, a high school student in public school 1420 who is attending a chemistry class could have a UI appropriate for performing chemistry experiments. Likewise, an airplane mechanic could also have a UI appropriate for repairing airplane engines. While both of these UIs would benefit from hands free, eyes out computing, the UI would be specifically and distinctively characterized for that particular system.

\* System characterization. When a computing system somehow infers a user's preferences and uses those preferences to design an optimal UI, the user preferences are considered to be system characterizations. These types of user preferences can be analyzed by the computing system over a specified period on time in which the computing system specifically detects patterns of use, learning style, level of expertise, and so on. Or, the user can play a game with the computing system that is specifically designed to detect these same characteristics.

\* Pre-configured. Some characterizations can be common and the UI can have a variety of pre-configured settings that the user can easily indicate to the UI. Pre-configured settings can include system settings and other popular user changes to default settings.

\* Remotely controlled. From time to time, it may be appropriate for someone or something other than the user to control the UI that is displayed.

#### EXAMPLE USER PREFERENCE CHARACTERIZATION VALUES

[00972] This UI characterization scale is enumerated. Some example values include:

- \* Self characterization
- \* Theme selection
- \* System characterization
- \* Pre-configured
- \* Remotely controlled

### THEME

[00973] A theme is a related set of measures of specific context elements, such as ambient temperature, current user task, and latitude, which reflect the context of the user. In other words, theme is a name collection of attributes, attribute values, and logic that relates these things. Typically, themes are associated with user goals, activities, or preferences. The context of the user includes:

- \* The user's mental state, emotional state, and physical or health condition.
- \* The user's setting, situation or physical environment. This includes factors external to the user that can be observed and/or manipulated by the user, such as the state of the user's computing system.
- \* The user's logical and data telecommunications environment (or "cyber-environment," including information such as email addresses, nearby telecommunications access such as cell sites, wireless computer ports, etc.).

[00974] Some examples of different themes include: home, work, school, and so on. Like user preferences, themes can be self characterized, system characterized, inferred, pre-configured, or remotely controlled.

### EXAMPLE THEME CHARACTERIZATION VALUES

[00975] This characteristic is enumerated. The following list contains example enumerated values for theme.

- \* No theme
- \* The user's theme is inferred.
- \* The user's theme is pre-configured.

- \* The user's theme is remotely controlled.
- \* The user's theme is self characterized.
- \* The user's theme is system characterized.

#### USER CHARACTERISTICS

[00976] User characteristics include:

- \* Emotional state
- \* Physical state
- \* Cognitive state
- \* Social state

#### EXAMPLE USER CHARACTERISTICS

#### CHARACTERIZATION VALUES

[00977] This UI characterization scale is enumerated. The following lists contain some of the enumerated values for each of the user characteristic qualities listed above.

- \* Emotional state.
  - \* Happiness
  - \* Sadness
  - \* Anger
  - \* Frustration
  - \* Confusion
- \* Physical state
  - \* Body
  - \* Biometrics
  - \* Posture
  - \* Motion
  - \* Physical Availability
    - \* Senses

- \* Eyes
- \* Ears
- \* Tactile
- \* Hands
- \* Nose
- \* Tongue
- \* Workload demands/effects
- \* Interaction with computer devices
- \* Interaction with people
- \* Physical Health
- \* Environment
  - \* Time/Space
  - \* Objects
  - \* Persons
    - \* Audience/Privacy Availability
      - \* Scope of Disclosure
      - \* Hardware affinity for privacy
      - \* Privacy indicator for user
      - \* Privacy indicator for public
      - \* Watching indicator
      - \* Being observed indicator
    - \* Ambient Interference
      - \* Visual
      - \* Audio
      - \* Tactile
- \* Location.
  - \* Place\_name

- \* Latitude
- \* Longitude
- \* Altitude
- \* Room
- \* Floor
- \* Building
- \* Address
- \* Street
- \* City
- \* County
- \* State
- \* Country
- \* Postal\_Code
- \* Physiology.
  - \* Pulse
  - \* Body\_temperature
  - \* Blood\_pressure
  - \* Respiration
- \* Activity
  - \* Driving
  - \* Eating
  - \* Running
  - \* Sleeping
  - \* Talking
  - \* Typing
  - \* Walking
- \*Cognitive state

- \* Meaning
- \* Cognition
  - \* Divided User Attention
  - \* Task Switching
  - \* Background Awareness
- \* Solitude
- \* Privacy
  - \* Desired Privacy
  - \* Perceived Privacy
- \* Social Context
- \* Affect
- \* Social state
  - \* Whether the user is alone or if others are present
  - \* Whether the user is being observed (e.g., by a camera)
  - \* The user's perceptions of the people around them and the user's perceptions of the intentions of the people that surround them.
  - \* The user's social role (e.g. they are a prisoner, they are a guard, they are a nurse, they are a teacher, they are a student, etc.)

#### COGNITIVE AVAILABILITY

[00978] There are three kinds of user tasks: focus, routine, and awareness and three main categories of user attention: background awareness, task switched attention, and parallel. Each type of task is associated with a different category of attention. Focus tasks require the highest amount of user attention and are typically associated with task-switched attention. Routine tasks require a minimal amount of user attention or a user's divided attention and are typically associated with parallel attention. Awareness tasks appeals to a user's precognitive state or attention and are typically associated with background awareness. When there is



an abrupt change in the sound, such as changing from a trickle to a waterfall, the user is notified of the change in activity.

## BACKGROUND AWARENESS

[00979] Background awareness is a non-focus output stimulus that allows the user to monitor information without devoting significant attention or cognition.

### EXAMPLE BACKGROUND AWARENESS

#### CHARACTERIZATION VALUES

[00980] This characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: the user has no awareness of the computing system/the user has background awareness of the computing system.

[00981] Using these values as scale endpoints, the following list is an example background awareness scale.

- \* No background awareness is available. A user's pre-cognitive state is unavailable.

- \* A user has enough background awareness available to the computing system to receive one type of feedback or status.

- \* A user has enough background awareness available to the computing system to receive more than one type of feedback, status and so on.

- \* A user's background awareness is fully available to the computing system. A user has enough background awareness available for the computing system such that they can perceive more than two types of feedback or status from the computing system.

### EXEMPLARY UI DESIGN IMPLEMENTATIONS FOR BACKGROUND AWARENESS

[00982] The following list contains examples of UI design implementations for how a computing system might respond to a change in background awareness.

\* If a user does not have any attention for the computing system, that implies that no input or output are needed.

\* If a user has enough background awareness available to receive one type of feedback, the UI might:

\* Present a single light in the peripheral vision of a user. For example, this light can represent the amount of battery power available to the computing system. As the battery life weakens, the light gets dimmer. If the battery is recharging, the light gets stronger.

\* If a user has enough background awareness available to receive more than one type of feedback, the UI might:

\* Present a single light in the peripheral vision of the user that signifies available battery power and the sound of water to represent data connectivity.

\* If a user has full background awareness, then the UI might:

\* Present a single light in the peripheral vision of the user that signifies available battery power, the sound of water that represents data connectivity, and pressure on the skin to represent the amount of memory available to the computing system.

## TASK SWITCHED ATTENTION

[00983] When the user is engaged in more than one focus task, the user's attention can be considered to be task switched.

## EXAMPLE TASK SWITCHED ATTENTION

### CHARACTERIZATION VALUES

[00984] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: the user does not have any attention for a focus task/the user has full attention for a focus task.

[00985] Using these characteristics as the scale endpoints, the following list is an example of a task switched attention scale.

\* A user does not have any attention for a focus task.

\* A user does not have enough attention to complete a simple focus task.

The time between focus tasks is long.

\* A user has enough attention to complete a simple focus task. The time between focus tasks is long.

\* A user does not have enough attention to complete a simple focus task.

The time between focus tasks is moderately long.

\* A user has enough attention to complete a simple focus task. The time between tasks is moderately long.

\* A user does not have enough attention to complete a simple focus task.

The time between focus tasks is short.

\* A user has enough attention to complete a simple focus task. The time between focus tasks is short.

\* A user does not have enough attention to complete a moderately complex focus task. The time between focus tasks is long.

\* A user has enough attention to complete a moderately complex focus task. The time between focus tasks is long.

\* A user does not have enough attention to complete a moderately complex focus task. The time between focus tasks is moderately long.

\* A user has enough attention to complete a moderately complex focus task. The time between tasks is moderately long.

\* A user does not have enough attention to complete a moderately complex focus task. The time between focus tasks is short.

\* A user has enough attention to complete a moderately complex focus task. The time between focus tasks is short.

\* A user does not have enough attention to complete a moderately complex focus task. The time between focus tasks is long.

\* A user has enough attention to complete a complex focus task. The time between focus tasks is long.

\* A user does not have enough attention to complete a complex focus task. The time between focus tasks is moderately long.

\* A user has enough attention to complete a complex focus task. The time between tasks is moderately long.

\* A user does not have enough attention to complete a complex focus task. The time between focus tasks is short.

\* A user has enough attention to complete a complex focus task. The time between focus tasks is short.

\* A user has enough attention to complete a very complex, multi-stage focus task before moving to a different focus task.

#### PARALLEL

[00986] Parallel attention can consist of focus tasks interspersed with routine tasks (focus task + routine task) or a series of routine tasks (routine task + routine task).

#### EXAMPLE PARALLEL ATTENTION

#### CHARACTERIZATION VALUES

[00987] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: the user does not have enough attention for a parallel task/the user has full attention for a parallel task.

[00988] Using these characteristics as scale endpoints, the following list is an example of a parallel attention scale.

\* A user has enough available attention for one routine task and that task is not with the computing system.

\* A user has enough available attention for one routine task and that task is with the computing system.

\* A user has enough attention to perform two routine tasks and at least of the routine tasks is with the computing system.

\* A user has enough attention to perform a focus task and a routine task. At least one of the tasks is with the computing system.

\* A user has enough attention to perform three or more parallel tasks and at least one of those tasks is in the computing system.

#### PHYSICAL AVAILABILITY

[00989] Physical availability is the degree to which a person is able to perceive and manipulate a device. For example, an airplane mechanic who is repairing an engine does not have hands available to input indications to the computing systems by using a keyboard.

#### LEARNING PROFILE

[00990] A user's learning style is based on their preference for sensory intake of information. That is, most users have a preference for which sense they use to assimilate new information.

#### EXAMPLE LEARNING STYLE CHARACTERIZATION VALUES

[00991] This characterization is enumerated. The following list is an example of learning style characterization values.

- \* Auditory
- \* Visual
- \* Tactile

#### EXEMPLARY UI DESIGN IMPLEMENTATION FOR LEARNING STYLE

[00992] The following list contains examples of UI design implementations for how the computing system might respond to a learning style.

- \* If a user is a auditory learner, the UI might:

- \* Present content to the user by using audio more frequently.
- \* Limit the amount of information presented to a user if there is a lot of ambient noise.
- \* If a user is a visual learner, the UI might:
  - \* Present content to the user in a visual format whenever possible.
  - \* Use different colors to group different concepts or ideas together.
  - \* Use illustrations, graphs, charts, and diagrams to demonstrate content when appropriate.
- \* If a user is a tactile learner, the UI might:
  - \* Present content to the user by using tactile output.
  - \* Increase the affordance of tactile methods of input (e.g. increase the affordance of keyboards).

## SOFTWARE ACCESSIBILITY

[00993] If an application requires a media-specific plug-in, and the user does not have a network connection, then a user might not be able to accomplish a task.

## EXAMPLE SOFTWARE ACCESSIBILITY

### CHARACTERIZATION VALUES

[00994] This characterization is enumerated. The following list is an example of software accessibility values.

- \* The computing system does not have access to software.
- \* The computing system has access to some of the local software resources.
- \* The computing system has access to all of the local software resources.
- \* The computing system has access to all of the local software resources and some of the remote software resources by availing itself to opportunistic user of software resources.

\* The computing system has access to all of the local software resources and all remote software resources by availing itself to the opportunistic user of software resources.

\* The computing system has access to all software resources that are local and remote.

### PERCEPTION OF SOLITUDE

[00995] Solitude is a user's desire for, and perception of, freedom from input. To meet a user's desire for solitude, the UI can do things like:

- \* Cancel unwanted ambient noise
- \* Block out human made symbols generated by other humans and machines

### EXAMPLE SOLITUDE CHARACTERIZATION VALUES

[00996] This characterization is scalar, with the minimum range being binary. Example binary values, or scalar endpoints, are: no solitude/complete solitude.

[00997] Using these characteristics as scale endpoints, the following list is an example of a solitude scale.

- \* No solitude
- \* Some solitude
- \* Complete solitude

### PRIVACY

[00998] Privacy is the quality or state of being apart from company or observation. It includes a user's trust of audience. For example, if a user doesn't want anyone to know that they are interacting with a computing system (such as a wearable computer), the preferred output device might be a head mounted display (HMD) and the preferred input device might be an eye-tracking device.

## HARDWARE AFFINITY FOR PRIVACY

[00999] Some hardware suits private interactions with a computing system more than others. For example, an HMD is a far more private output device than a desk monitor. Similarly, an earphone is more private than a speaker.

[001000] The UI should choose the correct input and output devices that are appropriate for the desired level of privacy for the user's current context and preferences.

## EXAMPLE PRIVACY CHARACTERIZATION VALUES

[001001] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: not private/private, public/not public, and public/private.

[001002] Using no privacy and fully private as the scale endpoints, the following list is an example privacy characterization scale.

- \* No privacy is needed for input or output interaction
- \* The input must be semi-private. The output does not need to be private.
- \* The input must be fully private. The output does not need to be private.
- \* The input must be fully private. The output must be semi-private.
- \* The input does not need to be private. The output must be fully private.
- \* The input does not need to be private. The output must be semi-private.
- \* The input must be semi-private. The output must be semi-private.
- \* The input and output interaction must be fully private.

[001003] \* Semi-private. The user and at least one other person can have access to or knowledge of the interaction between the user and the computing system.

[001004] \* Fully private. Only the user can have access to or knowledge of the interaction between the user and the computing system.



## EXEMPLARY UI DESIGN IMPLEMENTATION FOR PRIVACY

[001005] The following list contains examples of UI design implementations for how the computing system might respond to a change in task complexity.

- \* If no privacy is needed for input or output interaction:

- \* The UI is not restricted to any particular I/O device for presentation and interaction. For example, the UI could present content to the user through speakers on a large monitor in a busy office.

- \* If the input must be semi-private and if the output does not need to be private, the UI might:

- \* Encourage the user to use coded speech commands or use a keyboard if one is available. There are no restrictions on output presentation.

- \* If the input must be fully private and if the output does not need to be private, the UI might:

- \* Not allow speech commands. There are no restrictions on output presentation.

- \* If the input must be fully private and if the output needs to be semi-private, the UI might:

- \* Not allow speech commands (allow only keyboard commands). Not allow an LCD panel and use earphones to provide audio response to the user.

- \* If the output must be semi-private and if the input does not need to be private, the UI might:

- \* Restrict users to an HMD device and/or an earphone for output. There are no restrictions on input interaction.

- \* If the output must be semi-private and if the input does not need to be private, the UI might:

\* Restrict users to HMD devices, earphones, and/or LCD panels. There are no restrictions on input interaction.

\* If the input and output must be semi-private, the UI might:

\* Encourage the user to use coded speech commands and keyboard methods for input. Output may be restricted to HMD devices, earphones or LCD panels.

\* If the input and output interaction must be completely private, the UI might:

\* Not allow speech commands and encourage the user of keyboard methods of input. Output is restricted to HMD devices and/or earphones.

### USER EXPERTISE

[001006] As the user becomes more familiar with the computing system or the UI, they may be able to navigate through the UI more quickly. Various levels of user expertise can be accommodated. For example, instead of configuring all the settings to make an appointment, users can recite all the appropriate commands in the correct order to make an appointment. Or users might be able to use shortcuts to advance or move back to specific screens in the UI. Additionally, expert users may not need as many prompts as novice users. The UI should adapt to the expertise level of the user.

### EXAMPLE USER EXPERTISE CHARACTERIZATION VALUES

[001007] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: new user/not new user, not an expert user/expert user, new user/expert user, and novice/expert.

[001008] Using novice and expert as scale endpoints, the following list is an example user expertise scale.

\* The user is new to the computing system and to computing in general.

\* The user is new to the computing system and is an intermediate computer user.

\* The user is new to the computing system, but is an expert computer user.

\* The user is an intermediate user in the computing system.

\* The user is an expert user in the computing system.

#### EXEMPLARY UI DESIGN IMPLEMENTATION FOR USER EXPERTISE

[001009] The following are characteristics of an exemplary audio UI design for novice and expert computer users.

\* The computing system speaks a prompt to the user and waits for a response.

\* If the user responds in x seconds or less, then the user is an expert. The computing system gives the user prompts only.

\* If the user responds in >x seconds, then the user is a novice and the computing system begins enumerating the choices available.

[001010] This type of UI design works well when more than 1 user accesses the same computing system and the computing system and the users do not know if they are a novice or an expert.

#### LANGUAGE

[001011] User context may include language, as in the language they are currently speaking (e.g. English, German, Japanese, Spanish, etc.).

#### EXAMPLE LANGUAGE CHARACTERIZATION VALUES

[001012] This characteristic is enumerated. Example values include:

\* American English

\* British English

\* German

- \* Spanish
- \* Japanese
- \* Chinese
- \* Vietnamese
- \* Russian
- \* French

## COMPUTING SYSTEM

[001013] This section describes attributes associated with the computing system that may cause a UI to change.

[001014] Computing hardware capability

[001015] For purposes of user interfaces designs, there are four categories of hardware:

- \* Input/output devices
- \* Storage (e.g. RAM)
- \* Processing capabilities
- \* Power supply

[001016] The hardware discussed in this topic can be the hardware that is always available to the computing system. This type of hardware is usually local to the user. Or the hardware could sometimes be available to the computing system. When a computing system uses resources that are sometimes available to it, this can be called an opportunistic use of resources.

## STORAGE

[001017] Storage capacity refers to how much random access memory (RAM) is available to the computing system at any given moment. This number is not considered to be constant because the computing system might avail itself to the opportunistic use of memory.

[001018] Usually the user does not need to be aware of how much storage is available unless they are engaged in a task that might require more memory than to which they reliably have access. This might happen when the computing system engages in opportunistic use of memory. For example, if an in-motion user is engaged in a task that requires the opportunistic use of memory and that user decides to change location (e.g. they are moving from their vehicle to a utility pole where they must complete other tasks), the UI might warn the user that if they leave the current location, the computing system may not be able to complete the task or the task might not get completed as quickly.

### EXAMPLE STORAGE CHARACTERIZATION VALUES

[001019] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no RAM is available/all RAM is available.

[001020] Using no RAM is available and all RAM is available, the following table lists an example storage characterization scale.

Scale attribute	Implication
No RAM is available to the computing system	If no RAM is available, there is no UI available.—Or—There is no change to the UI.
Of the RAM available to the computing system, only the opportunistic use of RAM is available.	The UI is restricted to the opportunistic use of RAM.
Of the RAM that is available to the computing system, only the local RAM is accessible	The UI is restricted to using local RAM.
Of the RAM that is available to	The UI might warn the user that if

Scale attribute	Implication
the computing system, the local RAM is available and the user is about to lose opportunistic use of RAM.	they lose opportunistic use of memory, the computing system might not be able to complete the task, or the task might not be completed as quickly.
Of the total possible RAM available to the computing system, all of it is available.	If there is enough memory available to the computing system to fully function at a high level, the UI may not need to inform the user. If the user indicates to the computing system that they want a task completed that requires more memory, the UI might suggest that the user change locations to take advantage of additional opportunistic use of memory.

## PROCESSING CAPABILITIES

[001033] Processing capabilities fall into two general categories:

\* Speed. The processing speed of a computing system is measured in megahertz (MHz). Processing speed can be reflected as the rate of logic calculation and the rate of content delivery. The more processing power a computing system has, the faster it can calculate logic and deliver content to the user.

\* CPU usage. The degree of CPU usage does not affect the UI explicitly. With current UI design, if the CPU becomes too busy, the UI Typically “freezes” and the user is unable to interact with the computing system. If the CPU usage is too high, the UI will change to accommodate the CPU capabilities. For example, if

the processor cannot handle the demands, the UI can simplify to reduce demand on the processor.

## EXAMPLE PROCESSING CAPABILITY CHARACTERIZATION VALUES

[001034] This UI characterization is scalar, with the minimum range being binary. Example binary or scale endpoints are: no processing capability is available/all processing capability is available.

[001035] Using no processing capability is available and all processing capability as scale endpoints, the following table lists an example processing capability scale.

Scale attribute	Implication
No processing power is available to the computing system	There is no change to the UI.
The computing system has access to a slower speed CPU.	The UI might be audio or text only.
The computing system has access to a high speed CPU	The UI might choose to use video in the presentation instead of a still picture.
The computing system has access to and control of all processing power available to the computing system.	There are no restrictions on the UI based on processing power.

## POWER SUPPLY

[001046] There are two types of power supplies available to computing systems: alternating current (AC) and direct current (DC). Specific scale attributes for AC power supplies are represented by the extremes of the exemplary scale. However, if a user is connected to an AC power supply, it may be useful for the UI to warn an in-motion user when they're leaving an AC power supply. The user might need

to switch to a DC power supply if they wish to continue interacting with the computing system while in motion. However, the switch from AC to DC power should be an automatic function of the computing system and not a function of the UI.

[001047] On the other hand, many computing devices, such as wearable personal computers (WPCs), laptops, and PDAs, operate using a battery to enable the user to be mobile. As the battery power wanes, the UI might suggest the elimination of video presentations to extend battery life. Or the UI could display a VU meter that visually demonstrates the available battery power so the user can implement their preferences accordingly.

#### EXAMPLE POWER SUPPLY

##### CHARACTERIZATION VALUES

[001048] This task characterization is binary if the power supply is AC and scalar if the power supply is DC. Example binary values are: no power/full power. Example scale endpoints are: no power/all power.

[001049] Using no power and full power as scale endpoints, the following list is an example power supply scale.

- \* There is no power to the computing system.
- \* There is an imminent exhaustion of power to the computing system.
- \* There is an inadequate supply of power to the computing system.
- \* There is a limited, but potentially inadequate supply of power to the computing system.
- \* There is a limited but adequate power supply to the computing system.
- \* There is an unlimited supply of power to the computing system.



## EXEMPLARY UI DESIGN IMPLEMENTATIONS FOR POWER SUPPLY

[001050] The following list contains examples of UI design implementations for how the computing system might respond to a change in the power supply capacity.

- \* If there is minimal power remaining in a battery that is supporting a computing system, the UI might:

- \* Power down any visual presentation surfaces, such as an LCD.

- \* Use audio output only.

- \* If there is minimal power remaining in a battery and the UI is already audio-only, the UI might:

- \* Decrease the audio output volume.

- \* Decrease the number of speakers that receive the audio output or use earplugs only.

- \* Use mono versus stereo output.

- \* Decrease the number of confirmations to the user.

- \* If there is, for example, six hours of maximum-use battery life available and the computing system determines that the user not have access to a different power source for 8 hours, the UI might:

- \* Decrease the luminosity of any visual display by displaying line drawings instead of 3-dimensional illustrations.

- \* Change the chrominance from color to black and white.

- \* Refresh the visual display less often.

- \* Decrease the number of confirmations to the user.

- \* Use audio output only.

- \* Decrease the audio output volume.

## COMPUTING HARDWARE CHARACTERISTICS

[001051] The following is a list of some of the other hardware characteristics that may be influence what is an optimal UI design.

- \* Cost
- \* Waterproof
- \* Ruggedness
- \* Mobility

[001052] Again, there are other characteristics that could be added to this list. However, it is not possible to list all computing hardware attributes that might influence what is considered to be an optimal UI design until run time.

## BANDWIDTH

[001053] There are different types of bandwidth, for instance:

- \* Network bandwidth
- \* Inter-device bandwidth

## NETWORK BANDWIDTH

[001054] Network bandwidth is the computing system's ability to connect to other computing resources such as servers, computers, printers, and so on. A network can be a local area network (LAN), wide area network (WAN), peer-to-peer, and so on. For example, if the user's preferences are stored at a remote location and the computing system determines that the remote resources will not always be available, the system might cache the user's preferences locally to keep the UI consistent. As the cache may consume some of the available RAM resources, the UI might be restricted to simpler presentations, such as text or audio only.

[001055] If user preferences cannot be cached, then the UI might offer the user choices about what UI design families are available and the user can indicate their design family preference to the computing system.

## EXAMPLE NETWORK BANDWIDTH CHARACTERIZATION VALUES

[001056] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no network access/full network access.

[001057] Using no network access and full network access as scale endpoints, the following table lists an example network bandwidth scale.

Scale attribute	Implication
The computing system does not have a connection to network resources.	The UI is restricted to using local computing resources only. If user preferences are stored remotely, then the UI might not account for user preferences.
The computing system has an unstable connection to network resources.	The UI might warn the user that the connection to remote resources might be interrupted. The UI might ask the user if they want to cache appropriate information to accommodate for the unstable connection to network resources.
The computing system has a slow connection to network resources.	The UI might simplify, such as offer audio or text only, to accommodate for the slow connection. Or the computing system might cache appropriate data for the UI so the UI can always be optimized without restriction of the slow connection.

Scale attribute	Implication
The computing system has a high speed, yet limited (by time) access to network resources.	In the present moment, the UI does not have any restrictions based on access to network resources. If the computing system determines that it will lose a network connection, then the UI can warn the user and offer choices, such as does the user want to cache appropriate information, about what to do.
The computing system has a very high-speed connection to network resources.	There are no restrictions to the UI based on access to network resources. The UI can offer text, audio, video, haptic output, and so on.

#### INTER-DEVICE BANDWIDTH

[001070] Inter-device bandwidth is the ability of the devices that are local to the user to communicate with each other. Inter-device bandwidth can affect the UI in that if there is low inter-device bandwidth, then the computing system cannot compute logic and deliver content as quickly. Therefore, the UI design might be restricted to a simpler interaction and presentation, such as audio or text only. If bandwidth is optimal, then there are no restrictions on the UI based on bandwidth. For example, the UI might offer text, audio, and 3-D moving graphics if appropriate for the user's context.

## EXAMPLE INTER-DEVICE BANDWIDTH CHARACTERIZATION VALUES

[001071] This UI characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: no inter-device bandwidth/full inter-device bandwidth.

[001072] Using no inter-device bandwidth and full inter-device bandwidth as scale endpoints, the following table lists an example inter-device bandwidth scale.

Scale attribute	Implication
The computing system does not have inter-device connectivity.	Input and output is restricted to each of the disconnected devices. The UI is restricted to the capability of each device as a stand-alone device.
Some devices have connectivity and others do not.	It depends
The computing system has slow inter-device bandwidth.	The task that the user wants to complete might require more bandwidth that is available among devices. In this case, the UI can offer the user a choice. Does the user want to continue and encounter slow performance? Or, does the user want to acquire more bandwidth by moving to a different location and taking advantage of opportunistic use of bandwidth?
The computing system has fast inter-device bandwidth.	There are few, if any, restrictions on the interaction and

Scale attribute	Implication
	presentation between the user and the computing system. The UI sends a warning message only if there is not enough bandwidth between devices.
The computing system has very high-speed inter-device connectivity.	There are no restrictions on the UI based on inter-device connectivity.

### CONTEXT AVAILABILITY

- [001085] Context availability is related to whether the information about the model of the user context is accessible. If the information about the model of the context is intermittent, deemed inaccurate, and so on, then the computing system might not have access to the user's context.

### EXAMPLE CONTEXT AVAILABILITY

#### CHARACTERIZATION VALUES

- [001086] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: context not available/context available.
- [001087] Using context not available and context available as scale endpoints, the following list is an example context availability scale.
- [001088] \* No context is available to the computing system
  - [001089] \* Some of the user's context is available to the computing system.
  - [001090] \* A moderate amount of the user's context is available to the computing system.
  - [001091] \* Most of the user's context is available to the computing system.
  - [001092] \* All of the user's context is available to the computing system

## EXEMPLARY UI DESIGN FOR CONTEXT AVAILABILITY

[001093] The following list contains examples of UI design implementations for how the computing system might respond to a change in context availability.

\* If the information about the model of context is intermittent, deemed inaccurate, or otherwise unavailable, the UI might:

- \* Stay the same.
- \* Ask the user if the UI needs to change.
- \* Infer a UI from a previous pattern if the user's context history is available.
- \* Change the UI based on all other attributes except for user context (e.g. I/O device availability, privacy, task characteristics, etc.)
- \* Use a default UI.

## OPPORTUNISTIC USE OF RESOURCES

[001094] Some UI components, or other enabling UI content, may allow acquisition from outside sources. For example, if a person is using a wearable computer and they sit at a desk that has a monitor on it, the wearable computer might be able to use the desktop monitor as an output device.

## EXAMPLE OPPORTUNISTIC USE OF RESOURCES CHARACTERIZATION SCALE

[001095] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints, are: no opportunistic use of resources/use of all opportunistic resources.

[001096] Using these characteristics, the following list is an example of an opportunistic use of resources scale.

- \* The circumstances do not allow for the opportunistic use of resources in the computing system.

\* Of the resources available to the computing system, there is a possibility to make opportunistic use of resources.

\* Of the resources available to the computing system, the computing system can make opportunistic use of most of the resources..

\* Of the resources available to the computing system, all are accessible and available.

## CONTENT

[001097] Content is defined as information or data that is part of or provided by a task. Content, in contrast to UI elements, does not serve a specific role in the user/computer dialog. It provides informative or entertaining information to the user. It is not a control. For example a radio has controls (knobs, buttons) used to choose and format (tune a station, adjust the volume & tone) of broadcasted audio content.

[001098] Sometimes content has associated metadata, but it is not necessary.

[001099] Example content characterization values

[001100] This characterization is enumerated. Example values include:

- \* Quality

- \* Static/streamlined

- \* Passive/interactive

- \* Type

- \* Output device required

- \* Output device affinity

- \* Output device preference

- \* Rendering software

- \* Implicit. The computing system can use characteristics that can be inferred from the information itself, such as message characteristics for received messages.



- \* Source. A type or instance of carrier, media, channel or network path
- \* Destination. Address used to reach the user (e.g., a user typically has multiple address, phone numbers, etc.)
- \* Message content. (parseable or described in metadata)
- \* Data format type.
- \* Arrival time.
- \* Size.
- \* Previous messages. Inference based on examination of log of actions on similar messages.
- \* Explicit. Many message formats explicitly include message-characterizing information, which can provide additional filtering criteria.
  - \* Title.
  - \* Originator identification. (e.g., email author)
  - \* Origination date & time
  - \* Routing. (e.g., email often shows path through network routers)
  - \* Priority
  - \* Sensitivity. Security levels and permissions
  - \* Encryption type
  - \* File format. Might be indicated by file name extension
  - \* Language. May include preferred or required font or font type
  - \* Other recipients (e.g., email cc field)
  - \* Required software
  - \* Certification. A trusted indication that the offer characteristics are dependable and accurate.
  - \* Recommendations. Outside agencies can offer opinions on what information may be appropriate to a particular type of user or situation.

## SECURITY

[001101] Controlling security is controlling a user's access to resources and data available in a computing system. For example when a user logs on a network, they must supply a valid user name and password to gain access to resource on the network such as, applications, data, and so on.

[001102] In this sense, security is associated with the capability of a user or outside agencies in relation to a user's data or access to data, and does not specify what mechanisms are employed to assure the security.

[001103] Security mechanisms can also be separately and specifically enumerated with characterizing attributes.

[001104] Permission is related to security. Permission is the security authorization presented to outside people or agencies. This characterization could inform UI creation/selection by giving a distinct indication when the user is presented information that they have given permission to others to access.

### EXAMPLE SECURITY CHARACTERIZATION

#### VALUES

[001105] This characteristic is scalar, with the minimum range being binary. Example binary values, or scale endpoints are: no authorized user access/all user access, no authorized user access/public access, and no public access/public access.

[001106] Using no authorized user access and public access as scale endpoints, the following list is an example security scale.

- \* No authorized access.
- \* Single authorized user access.
- \* Authorized access to more than one person
- \* Authorized access for more than one group of people
- \* Public access

[001107] \* Single authorized user only access. The only person who has authorized access to the computing system is a specific user with valid user credentials.

[001108] \* Public access. There are no restrictions on who has access to the computing system. Anyone and everyone can access the computing system.

### TASK CHARACTERIZATIONS

[001109] A task is a user-perceived objective comprising steps. The topics in this section enumerate some of the important characteristics that can be used to describe tasks. In general, characterizations are needed only if they require a change in the UI design.

[001110] The topics in this section include examples of task characterizations, example characterization values, and in some cases, example UI designs or design characteristics.

### TASK LENGTH

[001111] Whether a task is short or long depends upon how long it takes a target user to complete the task. That is, a short task takes a lesser amount of time to complete than a long task. For example, a short task might be creating an appointment. A long task might be playing a game of chess.

### EXAMPLE TASK LENGTH CHARACTERIZATION VALUES

[001112] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: short/not short, long/not long, or short/long.

[001113] Using short/long as scale endpoints, the list is an example task length scale.

- \* The task is very short and can be completed in 30 seconds or less
- \* The task is moderately short and can be completed in 31-60 seconds.
- \* The task is short and can be completed in 61-90 seconds.

- \* The task is slightly long and can be completed in 91-300 seconds.
- \* The task is moderately long and can be completed in 301-1,200 seconds.
- \* The task is long and can be completed in 1,201-3,600 seconds.
- \* The task is very long and can be completed in 3,601 seconds or more.

### TASK COMPLEXITY

[001114] Task complexity is measured using the following criteria:

- \* Number of elements in the task. The greater the number of elements, the more likely the task is complex.

- \* Element interrelation. If the elements have a high degree of interrelation, then the more likely the task is complex.

- \* User knowledge of structure. If the structure, or relationships, between the elements in the task is unclear, then the more likely the task is considered to be complex.

[001115] If a task has a large number of highly interrelated elements and the relationship between the elements is not known to the user, then the task is considered to be complex. On the other hand, if there are a few elements in the task and their relationship is easily understood by the user, then the task is considered to be well-structured. Sometimes a well-structured task can also be considered simple.

### EXAMPLE TASK COMPLEXITY CHARACTERIZATION VALUES

[001116] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: simple/not simple, complex/not complex, simple/complex, well-structured/not well-structured, or well-structured/complex.

[001117] Using simple/complex as scale endpoints, the list is an example task complexity scale.



\* There is more than one complex task and each task is composed of 36-50 elements whose relationship is 40-60% understood by the user.

\* There is more than one very complex task and each part is composed of 51 or more elements whose relationship is 20-40% understood by the user.

### EXEMPLARY UI DESIGN IMPLEMENTATION FOR TASK COMPLEXITY

[001118] The following list contains examples of UI design implementations for how the computing system might respond to a change in task complexity.

- \* For a task that is long and simple (well-structured), the UI might:
  - \* Give prominence to information that could be used to complete the task.
  - \* Vary the text-to-speech output to keep the user's interest or attention.
- \* For a task that is short and simple, the UI might:

\* Optimize to receive input from the best device. That is, allow only input that is most convenient for the user to use at that particular moment.

\* If a visual presentation is used, such as an LCD panel or monitor, prominence may be implemented using visual presentation only.

- \* For a task that is long and complex, the UI might:
  - \* Increase the orientation to information and devices
  - \* Increase affordance to pause in the middle of a task. That is, make it easy for a user to stop in the middle of the task and then return to the task.

- \* For a task that is short and complex, the UI might:
  - \* Default to expert mode.
  - \* Suppress elements not involved in choices directly related to the current task.

- \* Change modality

## TASK FAMILIARITY

- [001119] Task familiarity is related to how well acquainted a user is with a particular task. If a user has never completed a specific task, they might benefit from more instruction from the computing environment than a user who completes the same task daily. For example, the first time a car rental associate rents a car to a consumer, the task is very unfamiliar. However, after about a month, the car rental associate is very familiar with renting cars to consumers.

### EXAMPLE TASK FAMILIARITY CHARACTERIZATION VALUES

- [001120] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: familiar/not familiar, not unfamiliar/unfamiliar, and unfamiliar/familiar.

- [001121] Using unfamiliar and familiar as scale endpoints, the list is an example task familiarity scale.

- \* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 1.

- \* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 2.

- \* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 3.

- \* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 4.

- \* On a scale of 1 to 5, where one is very unfamiliar and 5 is very familiar, the task familiarity rating is 5.

## EXEMPLARY UI DESIGN IMPLEMENTATION FOR TASK FAMILIARITY

[001122] The following list contains examples of UI design implementations for how the computing system might respond to a change in task familiarity.

- \* For a task that is unfamiliar, the UI might:
  - \* Increase task orientation to provide a high level schema for the task.
  - \* Offer detailed help.
  - \* Present the task in a greater number of steps.
  - \* Offer more detailed prompts.
  - \* Provide information in as many modalities as possible.
- \* For a task that is familiar, the UI might:
  - \* Decrease the affordances for help
  - \* Offer summary help
  - \* Offer terse prompts
  - \* Decrease the amount of detail given to the user
  - \* Use auto-prompt and auto-complete (that is, make suggestions based on past choices made by the user).
- \* The ability to barge ahead is available.
- \* Use user-preferred modalities.

## TASK SEQUENCE

[001123] A task can have steps that must be performed in a specific order. For example, if a user wants to place a phone call, the user must dial or send a phone number before they are connected to and can talk with another person. On the other hand, a task, such as searching the Internet for a specific topic, can have steps that do not have to be performed in a specific order.



## EXAMPLE TASK SEQUENCE CHARACTERIZATION VALUES

[001124] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: scripted/not scripted, nondeterministic/not nondeterministic, or scripted/nondeterministic.

[001125] Using scripted/nondeterministic as scale endpoints, the following list is an example task sequence scale.

- \* The each step in the task is completely scripted.
- \* The general order of the task is scripted. Some of the intermediary steps can be performed out of order.
- \* The first and last steps of the task are scripted. The remaining steps can be performed in any order.
- \* The steps in the task do not have to be performed in any order.

## EXEMPLARY UI DESIGN IMPLEMENTATION FOR TASK SEQUENCE

[001126] The following list contains examples of UI design implementations for how the computing system might respond to a change in task sequence.

- \* For a task that is scripted, the UI might:
  - \* Present only valid choices.
  - \* Present more information about a choice so a user can understand the choice thoroughly.
  - \* Decrease the prominence or affordance of navigational controls.
- \* For a task that is nondeterministic, the UI might:
  - \* Present a wider range of choices to the user.
  - \* Present information about the choices only upon request by the user.
  - \* Increase the prominence or affordance of navigational controls

## TASK INDEPENDENCE

- [001127] The UI can coach a user through a task or the user can complete the task without any assistance from the UI. For example, if a user is performing a safety check of an aircraft, the UI can coach the user about what questions to ask, what items to inspect, and so on. On the other hand, if the user is creating an appointment or driving home, they might not need input from the computing system about how to successfully achieve their objective.

### EXAMPLE TASK INDEPENDENCE CHARACTERIZATION VALUES

- [001128] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: coached/not coached, not independently executed/independently executed, or coached/independently executed.
- [001129] Using coached/independently executed as scale endpoints, the following list is an example task guidance scale.
- \* Each step in the task is completely scripted.
  - \* The general order of the task is scripted. Some of the intermediary steps can be performed out of order.
  - \* The first and last steps of the task are scripted. The remaining steps can be performed in any order.
  - \* The steps in the task do not have to be performed in any order.

## TASK CREATIVITY

- [001130] A formulaic task is a task in which the computing system can precisely instruct the user about how to perform the task. A creative task is a task in which the computing system can provide general instructions to the user, but the user uses their knowledge, experience, and/or creativity to complete the task. For example, the computing system can instruct the user about how to write a sonnet.

However, the user must ultimately decide if the combination of words is meaningful or poetic.

#### EXAMPLE TASK CREATIVITY CHARACTERIZATION VALUES

[001131] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints could be defined as formulaic/not formulaic, creative/not creative, or formulaic/creative.

[001132] Using formulaic and creative as scale endpoints, the following list is an example task creativity scale.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 1.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 2.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 3.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 4.

- \* On a scale of 1 to five, where 1 is formulaic and 5 is creative, the task creativity rating is 5.

#### SOFTWARE REQUIREMENTS

[001133] Tasks can be intimately related to software requirements. For example, a user cannot create a complicated database without software.

#### EXAMPLE SOFTWARE REQUIREMENTS CHARACTERIZATION VALUES

[001134] This task characterization is enumerated. Example values include:

- \* JPEG viewer

- \* PDF reader

- \* Microsoft Word
- \* Microsoft Access
- \* Microsoft Office
- \* Lotus Notes
- \* Windows NT 4.0
- \* Mac OS 10

### TASK PRIVACY

[001135] Task privacy is related to the quality or state of being apart from company or observation. Some tasks have a higher level of desired privacy than others. For example, calling a physician to receive medical test results has a higher level of privacy than making an appointment for a meeting with a co-worker.

### EXAMPLE TASK PRIVACY CHARACTERIZATION VALUES

[001136] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are: private/not private, public/not public , or private/public.

[001137] Using private/public as scale endpoints, the following table is an example task privacy scale.

- \* The task is not public. Anyone can have knowledge of the task.
- \* The task is semi-private. The user and at least one other person have knowledge of the task.
- \* The task is fully private. Only the user can have knowledge of the task.

### HARDWARE REQUIREMENTS

[001138] A task can have different hardware requirements. For example, talking on the phone requires audio input and output while entering information into a database has an affinity for a visual display surface and a keyboard.

## EXAMPLE HARDWARE REQUIREMENTS CHARACTERIZATION VALUES

[001139] This task characterization is enumerated. Example values include:

- \* 10 MB available of storage
- \* 1 hour of power supply
- \* A free USB connection

## TASK COLLABORATION

[001140] A task can be associated with a single user or more than one user. Most current computer-assisted tasks are designed as single-user tasks. Examples of collaborative computer-assisted tasks include participating in a multi-player video game or making a phone call.

## EXAMPLE TASK COLLABORATION CHARACTERIZATION VALUES

[001141] This task characterization is binary. Example binary values are single user/collaboration.

## TASK RELATION

[001142] A task can be associated with other tasks, people, applications, and so on. Or a task can stand alone on it's own.

## EXAMPLE TASK RELATION CHARACTERIZATION VALUES

[001143] This task characterization is binary. Example binary values are unrelated task/related task.

## TASK COMPLETION

[001144] There are some tasks that must be completed once they are started and others that do not have to be completed. For example, if a user is scuba diving and is using a computing system while completing the task of decompressing, it is essential that the task complete once it is started. To ensure the physical safety of

the user, the software must maintain continuous monitoring of the user's elapsed time, water pressure, and air supply pressure/quantity. The computing system instructs the user about when and how to safely decompress. If this task is stopped for any reason, the physical safety of the user could be compromised.

#### EXAMPLE TASK COMPLETION CHARACTERIZATION VALUES

[001145] This task characterization is enumerated. Example values are:

- \* Must be completed
- \* Does not have to be completed
- \* Can be paused
- \* Not known

#### TASK PRIORITY

[001146] Task priority is concerned with order. The order may refer to the order in which the steps in the task should be completed or order may refer to the order in which a series of tasks should be performed. This task characteristic is scalar. Tasks can be characterized with a priority scheme, such as (beginning at low priority) entertainment, convenience, economic/personal commitment, personal safety, personal safety and the safety of others. Task priority can be defined as giving one task preferential treatment over another. Task priority is relative to the user. For example, "all calls from mom" may be a high priority for one user, but not another user.

#### EXAMPLE TASK PRIVACY CHARACTERIZATION VALUES

[001147] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are no priority/high priority.

[001148] Using no priority and high priority as scale endpoints, the following list is an example task priority scale.

- \* The current task is not a priority. This task can be completed at any time.

\* The current task is a low priority. This task can wait to be completed until the highest priority, high priority, and moderately high priority tasks are completed.

\* The current task is moderately high priority. This task can wait to be completed until the highest priority and high priority tasks are addressed.

\* The current task is high priority. This task must be completed immediately after the highest priority task is addressed.

\* The current task is of the highest priority to the user. This task must be completed first.

### TASK IMPORTANCE

[001149] Task importance is the relative worth of a task to the user, other tasks, applications, and so on. Task importance is intrinsically associated with consequences. For example, a task has higher importance if very good or very bad consequences arise if the task is not addressed. If few consequences are associated with the task, then the task is of lower importance.

### EXAMPLE TASK IMPORTANCE CHARACTERIZATION VALUES

[001150] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are not important/very important.

[001151] Using not important and very important as scale endpoints, the following list is an example task importance scale.

\* The task is not important to the user. This task has an importance rating of "1."

\* The task is of slight importance to the user. This task has an importance rating of "2."

\* The task is of moderate importance to the user. This task has an importance rating of "3."

\* The task is of high importance to the user. This task has an importance rating of "4."

\* The task is of the highest importance to the user. This task has an importance rating of "5."

#### TASK URGENCY

[001152] Task urgency is related to how immediately a task should be addressed or completed. In other words, the task is time dependent. The sooner the task should be completed, the more urgent it is.

#### EXAMPLE TASK URGENCY CHARACTERIZATION VALUES

[001153] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are not urgent/very urgency.

[001154] Using not urgent and very urgent as scale endpoints, the following list is an example task urgency scale.

\* A task is not urgent. The urgency rating for this task is "1."

\* A task is slightly urgent. The urgency rating for this task is "2."

\* A task is moderately urgent. The urgency rating for this task is "3."

\* A task is urgent. The urgency rating for this task is "4."

\* A task is of the highest urgency and requires the user's immediate attention. The urgency rating for this task is "5."

#### EXEMPLARY UI DESIGN IMPLEMENTATION FOR TASK URGENCY

[001155] The following list contains examples of UI design implementations for how the computing system might respond to a change in task urgency.

\* If the task is not very urgent (e.g. a task urgency rating of 1, using the scale from the previous list), the UI might not indicate task urgency.



\* If the task is slightly urgent (e.g. a task urgency rating of 2, using the scale from the previous list), and if the user is using a head mounted display (HMD), the UI might blink a small light in the peripheral vision of the user.

\* If the task is moderately urgent (e.g. a task urgency rating of 3, using the scale from the previous list), and if the user is using an HMD, the UI might make the light that is blinking in the peripheral vision of the user blink at a faster rate.

\* If the task is urgent, (e.g. a task urgency rating of 4, using the scale from the previous list), and if the user is wearing an HMD, two small lights might blink at a very fast rate in the peripheral vision of the user.

\* If the task is very urgent, (e.g. a task urgency rating of 5, using the scale from the previous list), and if the user is wearing an HMD, three small lights might blink at a very fast rate in the peripheral vision of the user. In addition, a notification is sent to the user's direct line of sight that warns the user about the urgency of the task. An audio notification is also presented to the user.

### TASK CONCURRENCY

[001156] Mutually exclusive tasks are tasks that cannot be completed at the same time while concurrent tasks can be completed at the same time. For example, a user cannot interactively create a spreadsheet and a word processing document at the same time. These two tasks are mutually exclusive. However, a user can talk on the phone and create a spreadsheet at the same time.

### EXAMPLE TASK CONCURRENCY

#### CHARACTERIZATION VALUES

[001157] This task characterization is binary. Example binary values are mutually exclusive and concurrent.

### TASK CONTINUITY

[001158] Some tasks can have their continuity or uniformity broken without comprising the integrity of the task, while other cannot be interrupted without

compromising the outcome of the task. The degree to which a task is associated with saving or preserving human life is often associated with the degree to which it can be interrupted. For example, if a physician is performing heart surgery, their task of performing heart surgery is less interruptible than the task of making an appointment.

### EXAMPLE TASK CONTINUITY CHARACTERIZATION VALUES

[001159] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are interruptible/not interruptible or abort/pause.

[001160] Using interruptible/not interruptible as scale endpoints, the following list is an example task continuity scale.

- \* The task cannot be interrupted.
- \* The task can be interrupted for 5 seconds at a time or less.
- \* The task can be interrupted for 6-15 seconds at a time.
- \* The task can be interrupted for 16-30 seconds at a time.
- \* The task can be interrupted for 31-60 seconds at a time.
- \* The task can be interrupted for 61-90 seconds at a time.
- \* The task can be interrupted for 91-300 seconds at a time.
- \* The task can be interrupted for 301-1,200 seconds at a time.
- \* The task can be interrupted 1,201-3,600 seconds at a time.
- \* The task can be interrupted for 3,601 seconds or more at a time.
- \* The task can be interrupted for any length of time and for any frequency.

### COGNITIVE LOAD

[001161] Cognitive load is the degree to which working memory is engaged in processing information. The more working memory is used, the higher the

cognitive load. Cognitive load encompasses the following two facets: cognitive demand and cognitive availability.

[001162] Cognitive demand is the number of elements that a user processes simultaneously. To measure the user's cognitive load, the system can combine the following three metrics: number of elements, element interaction, and structure. Cognitive demand is increased by the number of elements intrinsic to the task. The higher the number of elements, the more likely the task is cognitively demanding. Second, cognitive demand is measured by the level of interrelation between the elements in the task. The higher the inter-relation between the elements, the more likely the task is cognitively demanding. Finally, cognitive load is measured by how well revealed the relationship between the elements is. If the structure of the elements is known to the user or if it's easily understood, then the cognitive demand of the task is reduced.

[001163] Cognitive availability is how much attention the user engages in during the computer-assisted task. Cognitive availability is composed of the following:

- \* Expertise. This includes schema and whether or not it is in long term memory
- \* The ability to extend short term memory.
- \* Distraction. A non-task cognitive demand.

#### HOW COGNITIVE LOAD RELATES TO OTHER ATTRIBUTES

[001164] Cognitive load relates to at least the following attributes:

- \* Learner expertise (novice/expert). Compared to novices, experts have an extensive schemata of a particular set of elements and have automaticity, the ability to automatically understand a class of elements while devoting little to no cognition to the classification. For example, a novice reader must examine every letter of the word that they're trying to read. On the other hand, an expert reader

- \* Task familiarity (unfamiliar/familiar). When a novice and an expert come across an unfamiliar task, each will handle it differently. An expert is likely to complete the task either more quickly or successfully because they access schemas that they already have and use those to solve the problem/understand the information. A novice may spend a lot of time developing a new schema to understand the information/solve the problem.

- \* Task complexity (simple/complex or well-structured/complex). A complex task is a task whose structure is not well-known. There are many elements in the task and the elements are highly interrelated. The opposite of a complex task is well-structured. An expert is well-equipped to deal with complex problems because they have developed habits and structures that can help them decompose and solve the problem.

\* Task length (short/long). This relates to how much a user has to retain in working memory.

\* Task creativity. (formulaic/creative) How well known is the structure of the interrelation between the elements?

### EXAMPLE COGNITIVE DEMAND

## CHARACTERIZATION VALUES

[001165] This task characterization is scalar, with the minimum range being binary. Example binary values or scale endpoints are cognitively undemanding/cognitively demanding.

## EXEMPLARY UI DESIGN IMPLEMENTATION FOR COGNITIVE LOAD

[001166] A UI design for cognitive load is influenced by a tasks intrinsic and extrinsic cognitive load. Intrinsic cognitive load is the innate complexity of the task and extrinsic cognitive load is how the information is presented. If the information is presented well (e.g. the schema of the interrelation between the elements is revealed), it reduces the overall cognitive load.

[001167] The following list contains examples of UI design implementations for how the computing system might respond to a change cognitive load.

- \* Present information to the user by using more than one channel. For example, present choices visually to the user, but use audio for prompts.

- \* Use a visual presentation to reveal the relationships between the elements. For example if a family tree is revealed, use colors and shapes to represent male and female members of the tree or shapes and colors can be used to represent different family units.

- \* Reduce the redundancy. For example, if the structure of the elements is revealed visually, do not use audio to explain the same structure to the user.

- \* Keep complementary or associated information together. For example, if creating a dialog box so a user can print, create a button that has the word "Print" on it instead of a dialog box that has a question "Do you want to print?" with a button with the work "OK" on it.

## TASK ALTERABILITY

[001168] Some task can be altered after they are completed while others cannot be changed. For example, if a user moves a file to the Recycle Bin, they can later retrieve the file. Thus, the task of moving the file to the Recycle Bin is alterable. However, if the user deletes the file from the Recycle Bin, they cannot retrieve it at a later time. In this situation, the task is irrevocable.

## EXAMPLE TASK ALTERABILITY CHARACTERIZATION VALUES

[001169] This task characterization is binary, with the minimum range being binary. Example binary values or scale endpoints are alterable/not alterable, irrevocable/revocable, or alterable/irrevocable.

## TASK CONTENT TYPE

[001170] This task characteristic describes the type of content to be used with the task. For example, text, audio, video, still pictures, and so on.

## EXAMPLE CONTENT TYPE CHARACTERISTICS VALUES

[001171] This task characterization is an enumeration. Some example values are:

- \* asp
- \* .jpeg
- \* .avi
- \* .jpg
- \* .bmp
- \* .jsp
- \* .gif
- \* .php
- \* .htm
- \* .txt
- \* .html
- \* .wav
- \* .doc
- \* .xls
- \* .mdb
- \* .vbs
- \* .mpg

[001172] Again, this list is meant to be illustrative, not exhaustive.

#### TASK TYPE

[001173] A task can be performed in many types of situations. For example, a task that is performed in an augmented reality setting might be presented differently to the user than the same task that is executed in a supplemental setting.

#### EXAMPLE TASK TYPE CHARACTERISTICS VALUES

[001174] This task characterization is an enumeration. Example values can include:

- \* Supplemental
- \* Augmentative
- \* Mediated

#### 3003: COMPARE UI DESIGNS WITH UI NEEDS

[001175] 3003 in figure 7 describes how to match an optimal UI characterization with a UI design characterization, as shown by the double-headed arrow in figure 1. First, the UI design characterizations are compared to the optimal UI characterizations (3004). This can be done, for example, by assembling the sets of characterizations into rows and columns of a look-up table. The following is a simple example of such a lookup table. The rows correspond to the UI design characterizations and the columns correspond to the UI needs characterizations.

[001176]

Design	Input device	Output device	Cognitive load	Privacy	Safety
A	1	2	3	4	
B	1	3	2	2	
C	2	1	1	1	

[001201] In figure 7, if there is not at least one match in the look-up table, then the closest match is chosen (3005). If there is more than one match, then the best

match is selected (3006). Once the match is made, it is sent to the computing system (3007).

#### 3004: ASSEMBLING UI DESIGNS AND UI NEEDS

[001202] As mentioned previously, this step of the process compares available UI design characterizations to UI needs characterizations. This can be done by matching XML metadata, numeric key metadata (such as values of a binary bit field), or assembling said metadata into rows and columns in a look-up table to determine if there is a match.

[001203] If there is a match, the request for that particular UI design is sent to the computing system and the UI changes.

#### 3005: CLOSEST MATCH

[001204] If there is no match for the current UI design, then the closest match is chosen. This section describes two ways to make the closest match:

- \* Using a weighted matching index.
- \* Creating explicit rules or logic

#### WEIGHTED MATCHING INDEX

[001205] In this embodiment, the optimal UI needs and UI design characterizations are assembled into a look-up table in 3004. If there is no match in the lookup table, then the characterizations of the current UI needs are weighted against the available UI designs and then the closest match is chosen. Figure 8 shows how this is done.

[001206] In figure 8, a weight is assigned to a particular characteristic or characteristics (4001, 4002, 4003, 4004). If the characterization in a design matches a UI design requirement, then the weighted number is added to the total. If a UI design characterization does not match a UI design requirement, then no value is added. For example, in the figure 8, the weighted matching index value for design A is "21." The logic used to determine this value is as follows:



If A(Input device) matches the first UI design requirement characterization value, then add 8. If it does not match, then do not add any value.

If A(Cognitive load) matches the cognitive load UI design requirement characterization value, then add 3. If there is no match, then do not add any value.

If A(Privacy) matches the Privacy UI design requirement characterization value, then add 10. If there is no match, then do not add any value.

[001207] However, there are times when some characteristics override all others. Figure 9 shows an example of such a situation.

[001208] In figure 9, even though attributes 5001, 5002, and 5003 do not match any 5available designs, 4004 matches the Safety characterization for design D. In this case, the logic used is as follows.

If A(Input device) matches the first UI design requirement characterization value, then add 8. If it does not match, then do not add any value.

If A(Cognitive load) matches the cognitive load UI design requirement characterization value, then add 3. If there is no match, then do not add any value.

If A(Privacy) matches the Privacy UI design requirement characterization value, then add 10. If there is no match, then do not add any value.

If A(Safety) matches the Safety UI design requirement characterization value, then choose design D.

[001209] The values for Input device, Cognitive load, Privacy, and Safety are determined by whether or not the characteristics are desirable, supplemental, or necessary. If a characteristic is necessary, then it gets a high weighted value. If a characteristic is desirable, then it gets next highest weighted value. If a characteristic is supplemental, then it gets the least amount of weight. In figure 8,

4004 is a necessary characteristic, 4001 and 4003 are desired characteristics, and 4002 is a supplemental characteristic.

### EXPLICIT RULES

[001210] Explicit rules can be implemented before (pre-matching logic), during (rules), or after (post-matching logic) the UI design choice is made.

### PRE-MATCHING LOGIC

[001211] The following is an example of pre-matching logic that can be applied to a look-up table to decrease the number of possible rows and/or columns in the table.

If personal risk is > moderate, then

If activity = driving, then choose design D, else

If activity = sitting, then choose design B, else

### RULES

[001212] The following is an example of an explicit rule that can be applied to a look-up table.

If Need = (Audio (Y) + Safety (high)), then choose only design B12.

Note: In this example, design B12 is the "Audio safety UI."

### POST-MATCHING LOGIC

[001213] At this step in the process, the computing system can verify with a user whether the choice is appropriate. This is optional. Example logic includes:

[001214] If the design has not been previously used, then verify with user.

### 3006: SELECTING THE BEST MATCH

[001215] There are two types of multiple matches. There are conditions in which more than one design is potentially suitable for a context characterization. Similarly, there are conditions in which a single UI design is suitable for more than one context characterization.

## UI FAMILY MATCH

- [001216] If a context characterization has more than one UI design match (e.g. there are multiple UI characterizations that match a context characterization), then the UI that is in the same UI family is chosen. UI family membership is part of the metadata that characterizes a UI design.

## NON UI FAMILY MATCH

- [001217] If none of the matches are in the same UI family, then the same mechanisms as described above can be used (weighted matching index, explicit logic, pre-matching logic, and post-matching logic).

- [001218] In figure 10, design D is the design of choice due to the following logic:

If A(Input device) matches the first UI design requirement characterization value, then add 8. If it does not match, then do not add any value.

If A(Cognitive load) matches the cognitive load UI design requirement characterization value, then add 3. If there is no match, then do not add any value.

If A(Privacy) matches the Privacy UI design requirement characterization value, then add 10. If there is no match, then do not add any value.

If A(Safety) matches the Safety UI design requirement characterization value, then choose design D, regardless of other characterization value matches.

[001219]

## DYNAMICALLY OPTIMIZING COMPUTER UIS

- [001220] By characterizing the function of user interface independently from its presentation and interaction with a broad set of attributes related to the changing needs of the user, in particularly to their changing contexts, a computer can make use of the various methods for optimizing a UI. These methods include the modification of:

Prominence – conspicuousness of a UI element.

Association – the indication of relationship between UI elements through similarity or grouping.

Metaphor

Sensory Analogy

Background Awareness

Invitation – Creating a sense of enticement or allurement to engage in interaction with a UI element(s).

Safety – A computer can enhance the safety of the user by either providing or emphasizing information that identifies real or potential danger or suggests a course of action that would allow the user to avoid danger, or a computer can suppress the presentation of information that may distract the user from safe actions, or it can offer modes of interaction that avoid either distraction or actual physical danger.

#### EXAMPLE CHARACTERISTICS OF AN EXAMPLE WPC

[001221] “Wearable” is a bit of a misnomer in that the defining characteristic of a WPC isn’t that it is worn or integrated into clothing, but that it travels with you at all times, is not removed or set down, and is considered by you and those around you as integral to your person, much as eyeglasses or a wristwatch or memories are. With such integration, wearable computers can truly become a component of you.

[001222] A wearable computer can also be distinguished by its ultimate promise: to serve as a capable, general-purpose computational platform which can, because it is always present, wholly integrate with your daily life.

[001223] The fuzzy description of a wearable computer is that it’s a computer that is always with you, is comfortable and easy to keep and use, and is as unobtrusive as clothing. However, this “smart clothing” description is unsatisfactory when

pushed in the details. A more specific description is that wearable computers have many of the following characteristics.

#### PRESENT AND OPERATIONAL IN ALL CIRCUMSTANCES

- [001224] The most distinguishing feature of a wearable is that it can be used while walking or otherwise moving around. You do not need to arrange yourself to suit the computer. Rather, the computer provides the means by which you can operate it regardless of circumstances. A wearable is designed to operate on you day and night, and no “place” is needed to set it up—neither a hand nor a flat surface. This distinguishes wearable computers from both desktop and laptop computers.

#### UNRESTRICTIVE

- [001225] A wearable is self-supporting on the body using some convenient means and works with you in all situations—walking, sitting, lying down. It doesn’t necessarily impinge on your life or what you’re doing. You can do other things while using it; for instance, you can walk to lunch while typing.

#### INTEGRAL

- [001226] A wearable is a part of “you,” like a wristwatch or eyeglasses or ears or thoughts. And like a wallet or watch, it is not separable or easily lost because it resides on you and effortlessly travels with you without your keeping track of it (as opposed to a briefcase). It is also integrated into your daily processes and can supplement thought as it takes place.

#### ALWAYS ON, ALERT, AND AVAILABLE

- [001227] By design, a wearable computer can be useful in whatever place you are in—it is always ready and responsive, reactive, proactive, and monitoring. It requires no setup time or manipulation to get started, unlike most pen-based personal digital assistants (PDAs) and laptops. (PDAs normally sit in a pocket and are only awakened when a task needs to be done; a laptop computer must be

opened up, switched on, and booted up before use.) A wearable is in continuous interaction with you, even though it may not be your primary focus at all times.

#### ABLE TO ATTRACT YOUR ATTENTION

[001228] A wearable can either make information available peripherally, or it can overtly interrupt you to gain your attention even when it's not actively being used. For example, if you want the computer to alert you when new e-mail arrives and to indicate its sender, the WPC can have a range of audible and visual means to communicate this depending on the urgency or importance of the e-mail, and on your willingness to deal with the notification at the time.

#### HOW A WEARABLE CHANGES THE WAY COMPUTERS FUNCTION

[001229] The promise of a wearable's unique characteristics make new uses of a computer inevitable.

#### THE COMPUTER CAN SENSE CONTEXT

[001230] Both interaction and information can be extremely contextual with a WPC. Given the right kind of sensors, the wearable can attend to (be aware of and draw input from) you and your environment. It could witness events around you, detect your circumstances or physical state (e.g., the level of ambient noise or privacy, whether you're sitting or standing), provide feedback about the environment (e.g., temperature, altitude), and adjust how it presents and receives information in keeping with your situation.

[001231] Always on and always sensing means a wearable might change which applications or UI elements it makes readily available as you move from work to home. Or it might tailor the UI and interaction to suit what's going on right now.

[001232] If it detects that you're flying, for instance, the wearable might automatically report your destination's local time and weather, track the status of your connecting flights, and help get you booked on another flight if your plane is going to be late. Similarly, if a wearable's sensors show that you're talking on the cell

phone, the WPC might automatically turn off audio interruptions and use only a head-mount display to alert you to incoming e-mails, calls, or information you have requested.

[001233] None of these uses are possible with a PDA or other computer system.

#### THE COMPUTER CAN SUGGEST AND/OR DIRECT

[001234] The better a WPC can sense context, the more appropriate and proactive its interaction can become for you. For instance, as you drive near your grocery store on the way home from work, your wearable might remind you that you should pick up cat food. This “eventing on context” gives the computer a whole new role in which it can suggest options and remind you of things like to putting out the trash on Tuesday or telling something to John as he walks into the room. You wouldn’t be able to do this with a desktop or laptop system.

[001235] A computer that is with you while you’re out in the world can also step you through processes and help troubleshoot problems within the very context in which they arise. This is different from a desktop system, which forces you to stay in its world, at its monitor, with your hands on its keyboard and mouse, printing out whatever instructions you may need offsite. The hands-free, always-with-you wearable can deliver procedures and instructions from any hard drive or web site at the very place where you’re faced with the problem. It can even direct you verbally or visually as you perform each step.

#### THE COMPUTER CAN AUGMENT INFORMATION, MEMORY, AND SENSES

[001236] Because a wearable computer can actively monitor, log, and preserve knowledge, it can have its own memories that you can rely on to augment your memory, intellect, or senses. For instance, its memory banks can help you recall where you parked the car at Disneyland, or replay the directions you asked for from the gas attendant. It might help you “sniff” carbon monoxide levels, see in

infrared or at night, and hear ultrahigh frequencies. When you're traveling in France, it might overlay English translations onto road signs.

## HOW A WEARABLE CHANGES THE WAY COMPUTERS AND PEOPLE INTERACT

[001237] Because a wearable computer is always around, always on, and always aware of you and your changing contexts, the WPC has the potential to become a working partner in almost any daily task. WPCs can prompt drastic shifts in how people interact with tools that were once viewed only as stationary, static devices.

## PEOPLE CAN BE IN TOUCH WITH THE WORLD IN WAYS NEVER BEFORE EXPERIENCED

[001238] A computer that can sense can be a digital mediator to the world around you. You can hear the pronunciation of unfamiliar words, call up a thesaurus or dictionary or translator or instructions, or pull up any Internet-based fact you need when you need it. Because a wearable can talk to any device within its range, it could annotate the world around you with relevant information. For example, it might overlay people's names as you meet them, provide menus of restaurants as you pass by, and list street names or historical buildings as you visit a new city. A wearable will be able to "sense across time" to provide an instant replay of recent events or audio, in case you missed what was said or done. And unlike smart phones which have to be turned on, a WPC can provide all of this information with a whisper or a keystroke anytime it's needed.

## THE COMPUTER CAN BE USED PERIPHERALLY THROUGHOUT THE DAY

[001239] A wearable PC turns computing into a secondary, not primary, activity. Unlike a desktop system that becomes your sole focus because it's time to sit down in front of it, a WPC takes on an ancillary, peripheral role by being always "awake" and available when it's needed, yet staying alert in the background when



you're busy with something else. Your interaction with a WPC is fluid and interruptible, allowing the computer to function as a supporting player throughout your day. This will make computer usage more incidental, with a get-in, get-out, and do-what-you-want focus.

## PEOPLE CAN ALTER THEIR COMPUTER INTERACTION BASED ON CONTEXT

[001240] WPCs imply that your use of, and interaction with, the computer can dramatically change from moment to moment based on your:

Physical ability to direct the system—You and the WPC will communicate differently based on what combination of your hands, ears, eyes, and voice is busy at the moment.

Physical (whole-body) activity—Your ability or willingness to direct the WPC may be altered by what action your whole body is doing, such as driving, walking, running, sitting, etc.

Mental attention or willingness to interact with the system (your cognitive availability)—How and whether you choose to communicate with the WPC may vary if you're concentrating on a difficult task, negotiating a contract, or shooting the breeze.

Context, task, need, or purpose—What you need the WPC for will vary by your current task or topic, such as if you're going to a meeting, in a meeting, driving around doing errands, or traveling on vacation.

Location—Both the content and nature of your WPC interaction can change as you move from an airplane in the morning, to an office during the day, to a restaurant for lunch, and then to a soccer game with the kids in the evening. They can also change even as you move through three-dimensional space.

Desire for privacy, perceived situational constraints—How you interact with the WPC is likely to change many times a day to accommodate the amount of

privacy you have or want, and whether you think using a WPC in a particular situation is socially acceptable.

## PEOPLE CAN INVEST THE COMPUTER WITH MORE ABOUT THEIR DAILY LIVES

[001241] Things originally considered trivial will now be input into and shared with the use of a computer. The issue of privacy both in interaction and content will become more important with a WPC, as well.

## EXAMPLE CHARACTERISTICS OF A DESIREABLE WPC UI OVERVIEW

1. Communicate the WPC's awareness of something to the user.
2. Receive acknowledgement or instructions from the user.

[001242] Just as the graphical user interface and mice made it easier to do certain things in a 2-D world of bitmap screens, so would a new UI make it easier to operate in the new settings demanded by wearable computing. Interfaces such as MS-Windows fail in a WPC setting. Based on the WPC's unique qualities and uses as defined in Section 2, the following are suggested capabilities of a successful wearable computer UI.

## A WPC UI SHOULD LET THE USER DIRECT THE SYSTEM UNDER ANY CIRCUMSTANCES.

[001243] Rationale Because the user's context, need for privacy, and physical and mental availability change all the time while using a WPC, the user should be able to communicate with the WPC using the most suitable input method of the moment. For instance, if he is driving or has his hands full or covered with grease, voice input would be preferable. However, if he's in a movie theater, on a subway, or in another public space where voice input may be inappropriate, he may prefer eye tracking or manual input.

[001244] In general, a UI's input system should accommodate minute-to-minute shifts in the user's:

Physical availability to direct the actions of the WPC, either with his hands (e.g., whether he has fine/gross motor control, or left/right/both/no hands free), voice, or other methods.

Mental availability to notice the WPC output and attend to or defer responding to it.

Desired privacy of the WPC interaction or content.

Context, task, or topic—that is, what his mind is working on at the moment.

[001245] Examples One way to direct a WPC under any circumstances is to allow the user to input in multiple ways, or modes (multi-modal input). The UI might offer all modes at once, or it might offer only the most appropriate modes for the context. In the former, the user would always be allowed to select the input mode that's appropriate to the context. In the latter, the UI would provide its best guess of input options and suppress the rest (e.g., if the room were dark, the UI might ignore taps on an unlighted keyboard but accept voice input).

[001246] Typical WPC multi-modal input methods could include touch pads, 1D and 2D pointing devices, voice, keyboard, virtual keyboard, handwriting recognition, gestures, eye tracking, and other tools.

#### A WPC UI SHOULD BE ABLE TO SENSE THE USER'S CONTEXT.

[001247] Rationale Ideally, a computer that is always on, always available, and not always the user's primary focus should be able to transcend all activities without the user always telling it what to do next. By "understanding" a context outside of itself, the WPC can change roles with the user and become an active support system. Doing so uses a level of awareness of the computer's outside surroundings that can drive and refine the appropriateness of WPC interactions, content, and WPC-initiated activities.

[001248] Current models of the UI between man and computer promote a master/slave relationship. A PC does the user's bidding and only "senses" the outside world through direct or indirect commands (via buttons, robotics, voice) from the user. Any input sensors that exist (e.g., cameras, microphones) merely reinforce this master/slave dynamic because they are controlled at the user's discretion. The computer is in essence deaf, dumb, blind, and non-sensing.

[001249] In the WPC world, the system has the potential to use computer-controlled (passive) sensors to hear, speak, see, and sense its own environment and the user's physical, mental, and contextual (content) states. By being aware of its own surroundings, the WPC can gather whatever information it wants (or thinks it needs) in order to appropriately respond to and serve its user.

[001250] The WPC UI should promote an exchange between man and machine that is a mix of active and passive interactions. As input is gathered, the UI should opportunistically generate a conceptual model of the world. It could use this model to make decisions in the moment (such as which output method is most appropriate or whether to send the person north or south when he's lost). It can also use the model to interpret and present information and choices to the user. Sensory information that is gathered but not relevant in the moment might also be accumulated for future action and knowledge.

[001251] Examples To become aware of its user and context, a WPC could accept input from automatic internal sensors or external devices, from the user with manual overrides (e.g., by speaking, "I'm now in the car"), or through other means.

[001252] An example of a WPC UI that mixes active and passive interaction would be when a person passes active information (choices) to the WPC while the WPC picks up on passive info (context, mood, temperature, etc). The WPC blends the active command with the passive information to build a conceptual model of

what's going on and what to do next. The computer then passes active information (such as a prompt or feedback) to the person and updates its conceptual model based on changes to its passive sensors.

#### A WPC UI SHOULD PROVIDE OUTPUT THAT IS APPROPRIATE TO THE USER'S CONTEXT.

[001253]      **Rationale** A WPC provides output to a user for three reasons. When it is being proactive, it initiates interaction by getting the user's attention (notification or system initiated activity). When it is being reactive, it provides a response to the user's input (feedback). When it is being passive or inactive, it could present the results of what it is sensing, such as temperature, date, or time (status).

[001254]      For an output to be appropriate to the context, the UI should:

Decide how and when it is best to communicate with the user. This should be based on his available attention and his ability/willingness to sense, direct, and process what the WPC is saying. For instance, the WPC might know to not provide audio messages while the user's on the phone.

Use a suitable output mechanism to minimize the disruption to the user and those around him. For instance, if the UI alerts a person about incoming mail, it might do so with only video in a noisy room, with only audio in a car, or with a blend of video and audio while the user is walking downtown.

Wait as necessary before interrupting the user to help the user appropriately shift focus. For instance, the WPC might wait until a phone call is completed before alerting him that e-mail has arrived.

[001255]      This is called having a scalable output.

[001256]      **Examples** One way to achieve scalable output is to use multiple output modes (multi-modal output). Typical WPC output modes could include video (monitors, lights, LEDs, flashes) through head-mounted and palm-top displays; audio (speech, beeps, buzzes, and similar sounds) through speakers or

earphones; and haptics (vibration or other physical stimulus) through pressure pads.

- [001257] Typical ways to address the appropriateness of the interaction include using and adjusting a suitable output mode for the user's location (such as automatically upping the volume on the earphone if in an airport), and waiting as necessary before interrupting the user (such as if he's in a meeting).

**A WPC UI SHOULD ACCOUNT FOR THE USER'S COGNITIVE  
AVAILABILITY.**

- [001258] **Rationale** A human being's capacity to process information changes throughout the day. Sometimes the WPC will be a person's primary focus; at others the system will be completely peripheral to his activities. Most often, the WPC will be used in divided-attention situations, with the user alternating between interacting with the WPC and interacting with the world around him. A WPC UI should help manage this varying cognitive availability in multiple ways.

**THE UI SHOULD ACCOMMODATE THE USER'S AVAILABLE  
ATTENTION TO ACKNOWLEDGE AND INTERPRET THE  
WPC.**

- [001259] **Rationale** An on-the-go WPC user prefers to spend the least amount of attention and mental effort trying to acknowledge and interpret what the WPC has told him. For instance, as the focus of a user's attention ebbs and flows, he might prefer to become aware of a notification, pause to instruct the WPC how to defer it, or turn his attention fully to accomplishing the related task.

- [001260] **Examples** Ways to accommodate the user's available attention include:  
Allow the user to set preferences of the intensity of an alert for a particular context.

Provide multiple and perhaps increasingly demanding output modes.

Make using the WPC a supportive, peripheral activity.

Build in shortcuts.

Use design elements such as consistency, color, prominence, positioning, size, movement, icons, and so on to make it clear what the WPC needs or expects.

#### THE UI SHOULD HELP THE USER MANAGE AND REDUCE THE MENTAL BURDEN OF USING THE WPC.

[001261]      Rationale    Because the user is likely to be multi-tasking with the WPC and the real world at the same time, the UI should seek to streamline processes so that the user can spend the least amount of time getting the system to do what he wants.

[001262]      Examples    Ways to reduce the burden of using WPC include:

Help chop work into manageable pieces.

Compartmentalize tasks.

Provide wizards to automate interactions.

Be proactive in providing alerts and information, so that the user can be reactive in dealing with them. (Reacting to something takes less mental energy than initiating it.)

#### THE UI SHOULD HELP THE USER RAPIDLY GROUND AND REGROUND WITH EACH USE OF THE WPC.

[001263]      Rationale    The UI should make it easy for a user to figure out what the WPC expects anytime he switches among contexts and tasks (grounding). It should also help him reestablish his mental connections, or return to a dropped task, after an interruption—such as when switching among applications, switching between use and non-use of the WPC, or switching among uses of the WPC in various contexts (regrounding).

[001264]      Examples    Ways to rapidly ground and reground include:

Use design devices such as prominence, consistency, and very little clutter.

Remember and redisplay the user's last WPC screen.

Keep a user log that he can be searched or backtracked.

Allow for thematic regrouping, so that the user will find the system and information as he last left them in a certain context. For instance, there could be themed settings for times when he is at home, at work, driving, doing a hobby, making home repairs, doing car maintenance, etc.

#### THE UI SHOULD PROMOTE THE QUICK CAPTURE AND DEFERRAL OF IDEAS AND ACTIONS FOR LATER PROCESSING.

[001265]      Rationale    A user prefers a low-effort, low-cognitive-load way to grab a fleeting thought as it comes, save it in whatever “raw” or natural format he wants, and then deal with it later when he is in a higher productivity mode.

[001266]      Examples    Ways to promote quick capture of information include:  
Record audio clips or .wav files and present them later as reminders.  
Take photos.  
Let the user capture back-of-the-napkin sketches.

#### A WPC UI SHOULD PRESENT ITS UNDERLYING CONCEPTUAL MODEL IN A MEANINGFUL WAY (OFFER CONSISTENT AFFORDANCE).

[001267]      Rationale    Affordance is the ability for something to inherently communicate how it is to be used. For instance, a door with a handle encourages a person to pull; one with only a metal plate encourages him to push. These are examples of affordance—the design of the tool itself, as much as possible, “affords” the information required to use the tool.



[001268] Far more so than for stationary computers, the interaction and functionality of a WPC should always be readily and naturally “grasped” if the UI is to support constant on-again, off-again use across many applications. This not only means that the UI elements should be self-evident in their purpose and functionality. It also means that the system should never leave the user guessing about what to do or say next—that is, the UI should expose, rather than conceal, as much as possible of how it “thinks.”

[001269] This underlying “conceptual model” (metaphor, structure, inherent “how-it-works”-ness) controls how every computer relates to the world. A UI that exposes its conceptual model speeds the learning curve, reinforces habit to reduce the cognitive load of using the WPC, and helps the user shortcut his way through the system without losing track of where his mind is in the real world around him. Input and output mechanisms that are this self-evident in how they are to be used are said to offer affordance. The goal of affordance is to have the user be able to say, “Oh, I know how to operate this thing,” when he is faced with something new.

[001270] Examples UI elements that replicate, as closely as possible, real-world experiences are most likely to be understood with very little training. For example, a two-state button (on/off) shouldn’t be used to make a person cycle through a three-state setting (low/medium/high). Instead, a dial, a series of radio buttons, an incremented slider bar, or some other mechanism should be used to imply more than an on/off choice.

[001271] Examples of how a UI can expose its underlying model include avoiding the use of hierarchical menus, using clear layman’s terms, building in idiomatic (metaphorical and consistent) operation, presenting all the major steps of a process at once to guide the user through, and making it clear which terms and commands the WPC expects to hear spoken, clicked, or input at any time.

## A WPC UI SHOULD HELP THE USER MANAGE HIS PRIVACY.

- [001272]      Rationale    Desktop monitors are usually configured to be private, and are treated as such by most people. However, because a WPC is around all the time, can log and output activity regardless of context, and becomes integrated with daily life, the issue of privacy becomes much more critical. At different times, a user might prefer either his content, his interaction with the WPC, or his information to stay private. Finding an unobserved spot to use a WPC is not always feasible—and having to do so is contrary to what a WPC is all about. A UI therefore should help the user continually manage the degree to which he wants privacy as situations change around him. In this context, there are four types of privacy the UI should account for.

### PRIVACY OF THE INTERACTION WITH WPC

- [001273]      Rationale    Because social mores or circumstances may dictate that interacting overtly with the WPC is unacceptable, a user might want to command the system without others knowing he's doing so. At the user's discretion, he should be able to make his interaction private or public, whether he's in a conference room, on a subway, or at a street corner.

- [001274]      Examples    Ways to achieve privacy of interaction include:

Use HMD and earpieces for output to the user.

Provide for non-voice input, such as eye-tracking or an unobtrusive keyboard or pointer.

### PRIVACY OF THE NATURE OF THE WPC INTERACTION

- [001275]      Rationale    Even if a person doesn't mind that others know he's using the WPC, he may not want others to eavesdrop on what he's trying to know, capture, call up, or retrieve, such as information, photos, e-mail, banking information, etc. The UI should support the desire to keep any combination of what the person is doing (e.g., making an appointment), saying (e.g., recording personal

information), or choosing (e.g., visiting a specific web site) secret from those around him.

- [001276]      Examples    Ways to achieve privacy of content of the interaction include:
- Use keyboard input with a head-mounted display (HMD).
  - Allow a user to speak his choices with codes instead of actual content (e.g., saying “3” then “5” instead of “Appointment” and “Fred Murtz” when scheduling a meeting).

#### PRIVACY OF THE WPC CONTENT

- [001277]      Rationale    Once a person has retrieved the information he wants (regardless of whether he cares if someone else knows what he’s calling up) he may not want others to actually hear or view the content. The UI should let him move into “secret mode” at any time.

- [001278]      Examples    Ways to achieve privacy of the content include:
- Provide a quick way for the user to switch from speakers or LCD panel output to a private-only mode, such as an HMD or earpiece.
  - Let the user set preferences that instruct the UI to switch automatically to private output based on content or context.

#### PRIVACY OF PERSONAL IDENTITY AND INFORMATION (SECURITY).

- [001279]      Rationale    A WPC is a logical place for a user to accumulate information about his identity, family, business, finances, and other information. The UI should provide an extremely secure, unforgettable identity that allows for anonymity when it’s desired, secure transactions, and protected, private information.

- [001280]      Examples    Ways to achieve security of identity include:
- Block another’s access to information that is within, or broadcast by, the WPC.

Selectively send WPC data only to specific people (such as the user's current location always to the spouse and family but not to anyone else).

#### A WPC UI SHOULD SCALE FROM PRIVATE TO COLLABORATIVE USE.

[001281]      **Rationale** Just as there are times when two or three people should huddle around a desktop system to share ideas, so a WPC user may want to shift from private-only viewing and interaction to collaborate with others. The UI should support ways to publicly share WPC content so that others can see what he sees, and perhaps also manipulate it.

[001282]      **Examples** Collaboration can be done by using a handheld monitor that both people can use at once or, if both people have WPCs, perhaps by wirelessly sharing the same monitor image on both HMDs. For collaborating with larger groups, the UI could support a way to transfer WPC information to a desktop or projection system, yet still let the user control what is viewed using standard WPC input methods.

#### A WPC UI SHOULD ACCEPT SPOKEN INPUT.

[001283]      **Rationale** A person should be able to command a WPC in any situation in which his hands are not free to manipulate a mouse, keyboard, or similar input device, such as when driving, carrying goods, or repairing an airplane engine. Using voice to control the WPC is a natural choice for almost all hands-busy situations. The WPC UI should therefore support and utilize a speech recognition system that understands what a user will say to it.

[001284]      **Examples** Computer-based speech recognition capability can range from recognizing everything that a person can say (understanding natural language), to recognizing words and phrases from a large predefined vocabularies (such as thousands of words), to recognizing only a few dozen select words at a time (very

limited vocabulary). Another level of speech recognition involves being able to also understand the way (tone) in which something is said.

#### A WPC UI SHOULD SUPPORT TEXT INPUT METHODS.

[001285]       Rationale   A user is likely to want to capture brief text strings that the WPC has never seen before, such as people's names and URLs. For this reason, a UI should allow the user to accurately save, input, and/or select custom textual information. This capability should span multiple input modes, in keeping with the WPC's value as a hands-free, use-anywhere device.

[001286]       Examples   Accurate text input can be provided through a keyboard, virtual keyboard, handwriting recognition, voice spelling, and similar mechanisms.

#### A WPC UI SHOULD SUPPORT MULTIPLE KINDS OF VOICE INPUT.

[001287]       Rationale   An ordinary computer microphone cannot discern between when someone is talking to the system or to someone else in the room. A microphone-equipped WPC is supposed to be able to understand and recognize this subtlety and process a user's voice input in several listening modes, including:

          Voice commands—the computer instantly responds to instructions given without a pointer or keyboard.

          Phone conversation—the system recognizes when its user's voice is directed to a phone instead of to it.

          Recorded voice—the computer creates a .wav file or similar image of the sound on demand; this could be used with phone input and output.

          Dictation to transcript—the system converts speech into ASCII on the fly.

          Dictation to text box—in this special case of transcription, the computer accepts words from a constrained vocabulary and converts them to ASCII to insert into a given field, such as saying "December 16" and having it show up on a Date field on a form.

Dictation training—the system learns an individual’s idiosyncratic pronunciation of words.

Silence—the system leaves its microphone on and awaits instructions; it may passively indicate volume.

Mode switch—the system understands that the user wants to switch between listening modes, such as with, “Computer <state|context|function|user-defined>” or “Computer, end transcription.”

Speaker differentiation—the computer recognizes its own user’s voice, so that when someone else gives a command either deliberately or in the background, the system ignores it.

[001288] The UI should manage each type of voice transition fluidly and (preferably) in a hands-free manner.

[001289] Examples Using a push-to-talk button can alert the system when it is being addressed, and user settings or preferences can make it clear when to record or not record, when to listen or not listen, and how to respond in each case.

#### A WPC UI SHOULD WORK WITH MULTIPLE WPC APPLICATIONS.

[001290] Rationale The value of a WPC is its ability to be used in multiple ways and for multiple purposes throughout the day. Related tasks will generally be grouped into one or more WPC applications that can help organize and simplify tasks, as well as help reduce the cognitive load of using the WPC.

[001291] Examples Single and group applications for WPCs are virtually limitless. Examples include forms creation, web linking, online readers, e-mail, phone, location (GPS), a datebook, a contacts book, camera, scanning tools, video and voiceover input, and tools to capture scrawled pictures.

## A WPC UI SHOULD ALLOW AND ASSIST WITH MULTITASKING AND SWITCHING AMONG WPC APPLICATIONS.

[001292]      Rationale    Many times a day, a WPC user will require more than one WPC application running at the same time to complete a task. For instance, when making an appointment with someone, a user might use an address book application to retrieve his photo and contact information; use a phone application to call him up; use a journal application to look up the information they were last talking about; use a voice recorder application to capture the audio of a phone call; use a note-taking application to scribble down notes and share with someone else who's standing by; use an e-mail application to attach the scribble to an e-mail; and use a to-do application to check off the phone call as a completed task and flag another task for follow-up. In all cases of cross-application work, the UI should help the user keep track of where he is, where he's been, and how to get where he wants to go.

[001293]      Examples    Ways that a UI could help the user keep track of these applications include:

Use icons that indicate which application(s) are on and which one is active.

Include logging methods to help the user back-track to the place where he left off from application to application.

Provide tools to jump ad hoc between applications at any time.

## A WPC UI SHOULD BE EXTENSIBLE TO FUTURE TECHNOLOGIES.

[001294]      Rationale    As the wearable gains popularity, WPC uses that are unheard of today will become standard tomorrow. For this reason, the UI should be designed so that it is open enough to fold in new functionality in a consistent manner. Such new functionality might include enriched methods for gaining the user's attention, improvements to the WPC's context-awareness sensors, and new applications.

[001295]        Examples    Ways to make sure a UI is extensible include utilizing and building from currently accepted standards, or coding with an open or module-based architecture.

## DETAILS OF AN EXAMPLE UI

### OVERVIEW OF EXAMPLE WPC SOFTWARE AND TOOLS

[001296]        Five example types of products for WPCs:

User Interface (UI)—what the user interacts with. The UI enables the user and the WPC to hold a dialog—that is, to exchange input and output. It amends and facilitates this conversation. The UI solves the need for a WPC that a user can command and interact with.

Applets (many may be developed by third parties)—the WPC applications that run within the interface. Applets allow the user to accomplish specific tasks with a WPC, such as make a phone call or look up an online manual. They provide a means to input information that's relevant to the task at hand, and facilitate the tasks' completion. Applets solve the need for a WPC that can be useful in real-world situations.

Characterization Module (CM)—an architectural framework that allows awareness to be added to a WPC as WPC use evolves. In particular, the CM tells the WPC about the user's context, such as his physical, environmental, social, mental, or emotional state. It senses the external world, provides status or reporting to the UI, and facilitates UI conceptual models. The CM solves the need for a WPC that can sense the world around it.

Developer tools—software kits designed to help others develop compatible software. These comprise SDKs, sample software, and other instructional materials for use by developers and OEMs. Developer tools solve the need for how others can design applications and sensors that a WPC can use.



Portal—a future web site where people can find WPC Applets, upgrades, and new WPC services from developers. The Portal solves the need for keeping developers, users, and OEMs up to date on WPC-related information and software.

#### THE EXAMPLE UI WILL MANAGE INPUT AND OUTPUT INDEPENDENTLY OF APPLLET FUNCTIONALITY.

Supported UI Requirement: 0 A WPC UI should let the user direct the system under any circumstance.

Supported UI Requirement: 0 A WPC UI should provide output that is appropriate to the user's context.

Supported UI Requirement: 0 A WPC UI should allow and assist with multitasking and switching among WPC applications.

Supported UI Requirement: 0 A WPC UI should be extensible to future technologies.

[001297]      Rationale    For a WPC to achieve its ultimate value throughout the day, the UI should always reveal the workings of the system, what it's looking for from the user, and what the person can do with it—all suitable to the context. Moreover, how the system handles these three facets should be consistent, so that someone doesn't have to learn a whole new WPC mechanism with every Applet or input method.

[001298]      To achieve these goals, the Example UI splits the WPC experience into three interrelated facets:

Presentation—what the user sees, hears, and senses from the WPC (WPC output). Presentation determines what the UI and the Applets look like and how intuitively and quickly they can be understood. Presentation can be achieved through audio, video, physical (haptics), or some combination.



THE EXAMPLE UI WILL PRESENT ALL AVAILABLE INPUT OPTIONS  
AT ONCE.

Supported UI Requirement: 0 The UI should let the user direct the system under any circumstances.

[001301]      Rationale    Current sensor technology will make it very difficult for the WPC to determine the user's context enough to present and accept only the kind of input that is appropriate to the situation. Rather than have the UI make an error of omission of input methods, it will present all available input options at once and expect the user to choose which one he wants to use.

[001302]      Ideas and implications    One way around the all-or-nothing input options is to have the user be able to set thematic preferences, such as "When I'm in the car, don't bother to activate the keyboard."

THE EXAMPLE UI WILL ALWAYS MAKE ALL INPUT OPTIONS  
OBVIOUS.

[001303]      Rationale    An overriding goal for the Example UI is to make it fast and easy for the user to get in and out of a WPC interaction. As the UI prompts him for decisions and input, the user should be able to tell the following from the UI:

When voice, keyboard, stylus, or whatever other input option can be applied.

Which words the WPC will respond to verbally.

What keyboard and mouse/stylus actions are equivalent to voice.

[001304]      Ideas and implications    Visual can be the default (provides parallel input for faster interaction), but the user should be able to switch to audio (provides serial input for slower interaction) if appropriate. The UI should provide multiple and consistent mechanism(s) to enter new terms, names, an URLs. For this purpose, the UI should make it clear that the WPC supports: keyboard input, virtual keyboard input, voice spelling, and handwriting recognition (rudimentary).

The methods for entering new names, etc. should be consistently available and consistently operated.

THE EXAMPLE UI WILL BE AS PROACTIVE AS POSSIBLE WITH  
NOTIFICATION CUES.

Supported UI Requirement: 0 The UI should support the user's cognitive availability.

[001305] Rationale A WPC that can detect a user's context can play a significant role if it can proactively notify the user when things happen and prompt him for decisions and input. Presenting information and staging interactions so that the user can be reactive in handling them lowers the cognitive load required and makes the WPC less of a burden to use. The level of this proactivity may be limited by current sensor technology. To be proactive, the UI's notifications and prompts should:

Be a supportive, peripheral activity that is appropriate to the context—e.g., no audio messages while the user's on the phone, or perhaps it should even wait until the phone call is completed before alerting him.

Use a suitable output mechanism—e.g., into ear or eye, preferably depending on where user is at the moment (in car, at home, at office, in airplane).

Wait as necessary before interrupting the user—e.g., if he's on the phone. The user's ability to devote divided or undivided attention to the WPC interaction determines whether he is interruptible.

THE EXAMPLE UI WILL ALLOW 1-D AND 2-D INPUT, BUT NOT  
DEPEND TOO HEAVILY ON IT.

Supported UI Requirement: 0 The UI should let the user direct the system under any circumstances.

Supported UI Requirement: 0 The UI should provide output that is appropriate to the user's context.

Supported UI Requirement: 0 The UI should accept spoken input.

Supported UI Requirement: 0 The UI should account for the available attention to acknowledge the WPC.

[001306]      Rationale      When the user needs hands-on input such as typing or mousing, the WPC should support standard pointing and keyboard modes. However, the WPC should also be able to be used in hands-busy and eyes-busy circumstances, which demands the use of speech input and output. However, a two-dimensional, pointer-driven UI (such as most current WIMP applications) doesn't always translate well to voice-only commands. For instance, a user should not be forced into a complicated description of where to place the pointer before selecting something, nor should he be expected to use vocal variances (e.g., trills to grunts) to tell the cursor to move up and down or left and right. The Example UI will depend more on direct voice input/output and less heavily on 2-D output and input that can't be readily translated to voice.

[001307]      Ideas and implications      Exposing items as a list lets users choose what they want either verbally or with a pointer.

THE EXAMPLE UI WILL SCALE WITH THE USER'S EXPERTISE.

Supported UI Requirement: 0 The UI should let the user direct the system under any circumstances.

Supported UI Requirement: 0 The UI should account for the available attention to acknowledge the WPC.

[001308]      Rationale      Scale on expertise - shortcuts/post processing assists with cognitive load

THE EXAMPLE UI WILL SURFACE ITS BEST GUESS ABOUT THE USER'S CONTEXT.

Supported UI Requirement: 0 The UI should provide output that is appropriate to the user's context.

Supported UI Requirement: 0 The UI should help the user manage and reduce the mental burden of interacting with the WPC.

Supported UI Requirement: 0 The UI should help the user rapidly ground and reground with each use of the WPC.

[001309]      Rationale    Building from Characterization Module sensors, the UI should surface its best guess of the user's ability to direct, sense, and think or process at any time. Methods to set attributes could be both fine grained ("My eyes are not available now," which could set the system to use the earpiece) and thematic ("I am driving now," which could set information context plus eyes and hands not available). Eyes and ears can be available in diminishing capacity. Generally a person can't have fine and gross motor control simultaneously.

[001310]      Ideas and implications    From a UI standpoint, awareness could be manifest by changing the display to reveal what the system thinks is the context, yet still allow the user to change that context back to where he last was, or to something else altogether.

#### THE EXAMPLE UI WILL REVEAL ALL OF AN APPLET'S AVAILABLE OPTIONS AT ALL TIMES.

Supported UI Requirement: 0 The UI should help the user manage and reduce the mental burden of interacting with the WPC.

Supported UI Requirement: 0 The UI should help the user rapidly ground and reground with each use of the WPC.

Supported UI Requirement: 0 The UI should present its underlying conceptual model in a meaningful way (offer affordance).

[001311]      Rationale    Rather than bury commands in multiple menus that force the user to pay close attention to learning and interacting with the WPC, the Example UI should expose all available user options all the time for each active Applet.

This way, the user can see all of his choices (e.g., available tasks, not all data items such as names or addresses) at once.

#### THE EXAMPLE UI WILL NEVER BE A BLANK SLATE.

Supported UI Requirement: 0 The UI should help the user manage and reduce the mental burden of interacting with the WPC.

Supported UI Requirement: 0 The UI should help the user rapidly ground and reground with each use of the WPC.

[001312]      Rationale    By definition, a WPC that is context-aware should always be able to show information that is trenchant to the current circumstances. Even in “idle” mode, there is no reason for the WPC to be a blank slate. A continuously context-sensitive UI can help the user quickly ground when using the system, reduce the mental attention needed to use it, and depend on the WPC to provide just the right kind of information at just the right time.

[001313]      Ideas and implications    If the system is idle, it might display something different by default if the person is at home vs. if he’s at the office. Similarly, if the person actively has an Applet running (such as a To Do list), what the UI shows could vary by where the user is—on the way home past Safeway or in an office.

#### THE EXAMPLE UI WILL BE CONSISTENT.

Supported UI Requirement: 0 The UI should help the user manage and reduce the mental burden of interacting with the WPC.

Supported UI Requirement: 0 The UI should help the user rapidly ground and reground with each use of the WPC.

Supported UI Requirement: 0 A WPC UI should allow and assist with multitasking and switching among WPC applications.

[001314]      Rationale    Throughout the day, a user’s interaction with the WPC will occur amidst many distractions, in differing contexts, and across multiple related WPC applications. For this reason, the UI should provide fundamentally the same

kind of interaction for every similar kind of input. For instance, what works for a voice command in one situation should work for a voice command in a similar situation. This consistency enables the user to:

Quickly grasp how to first use the WPC and what it expects at any given time.

Minimize his interaction time with the WPC and gain faster, more accurate results.

Reliably extrapolate how to use new WPC functionality as it becomes available.

[001315] Ideas and implications A consistent user interface should:

Make all applications operate through the same modes in the same way (such as through consistent voice or keyboard commands).

Make text input consistently available and operated.

Make it clear at all times which part of the UI the user is supposed to interact with (vs., say, which parts he only has to read).

Use standard formats for time, dates, GPS/location, etc. so that many Applets can use them.

THE EXAMPLE UI WILL BE CONCISE AND UNCLUTTERED.

Supported UI Requirement: 0 The UI should help the user manage and reduce the mental burden of interacting with the WPC.

Supported UI Requirement: 0 The UI should help the user rapidly ground and reground with each use of the WPC.

[001316] Rationale A WPC will often be used when attention to visual detail in the UI is unrealistic, such as while driving or in a meeting. The UI should therefore be concise, offering just enough information at all times. What is "just enough" should also be tempered by how much can be absorbed at one time. To promote the get-



in-and-get-out nature of a WPC, the Example UI should also be designed with as little visual clutter as possible.

- [001317] Ideas and implications In particular, the UI should display ear or eye output without obstructing anything else.

THE EXAMPLE UI WILL GUIDE THE USER TO WHAT IS MOST IMPORTANT.

Supported UI Requirement: 0 The UI should help the user manage and reduce the mental burden of interacting with the WPC.

Supported UI Requirement: 0 The UI should help the user rapidly ground and reground with each use of the WPC.

Supported UI Requirement: 0 A WPC UI should allow and assist with multitasking and switching among WPC applications.

- [001318] Rationale A fully context-aware WPC would be able to detect and keep track of a user's priorities, and constantly present information that's relevant to his content, purpose, environment, or level of urgency. When deciding what to present and when to present it, the UI should be able to guide the customer to what is most important to deal with at any given moment.

- [001319] Ideas and implications This can be done through UI design techniques such as prominence, color, and motion.

THE EXAMPLE UI WILL GUIDE THE USER ABOUT WHAT TO DO NEXT.

Supported UI Requirement: 0 The UI should help the user manage and reduce the mental burden of interacting with the WPC.

Supported UI Requirement: 0 The UI should help the user rapidly ground and reground with each use of the WPC.

Supported UI Requirement: 0 The UI should present its underlying conceptual model in a meaningful way (offer affordance).

Supported UI Requirement: 0 A WPC UI should allow and assist with multitasking and switching among WPC applications.

[001320]      Rationale    As much as possible, the Example UI should assist the user so as to minimize the time to understand what to do, how to do it, and how to process what doing it has accomplished. The UI should provide a way for the user to know that a command is available and that his input has been received correctly. It should also help him reload dropped information and reground to a dropped task after or during an interruption in the task.

[001321]      Ideas and implications    A popular approach is to make everything that the user can do be visible and to have UI constrain what the WPC will recognize. For instance, text that is in gold can be said aloud, but text in the bouncing ball list exposes what to expect next in an Applet's process in a linear, language-oriented way. Incremental typing letters filters a list down.

#### THE EXAMPLE UI WILL ALWAYS REVEAL THE USER'S PLACE IN THE SYSTEM.

Supported UI Requirement: 0 The UI should allow and assist with multitasking and switching among WPC applications.

Supported UI Requirement: 0 The UI should help the user manage and reduce the mental burden of interacting with the WPC.

Supported UI Requirement: 0 The UI should help the user rapidly ground and reground with each use of the WPC.

[001322]      Rationale    Because the user's focus and attention will often shift back and forth between the WPC and his surroundings, the UI should clearly show him where he is within the UI at all times (e.g., "I'm currently operating the Calendar and am this far along in it"). This means letting him switch among Applets easily without losing track of where he's been, as well as determining and returning to his previous state if he is doing "nested" work among several Applets.

[001323] Ideas and implications Orienting can be done through UI design techniques such as color, icons, banners, title bars, etc.

#### THE EXAMPLE UI WILL USE A FINITE SPOKEN VOCABULARY

Supported UI Requirement: 0 The UI should help the user manage and reduce the mental burden of interacting with the WPC.

Supported UI Requirement: 0 The UI should accept spoken input.

[001324] Rationale The current state of the art for speech recognition does not allow for natural language or large vocabularies. The dialog between computers and people is not like person-to-person conversation. People don't speak the same way in all settings, and the user may not be able to train the WPC. Meaning, tone, and nuance are difficult to capture accurately in a person-to-computer interaction. Voice systems are by nature linear and tedious because all interactions should be serial. Ambient sounds and quality of voice pickup dramatically affect the robustness of speech recognition programs. To succeed, the Example UI should not require a large vocabulary to use. However, the speech should be as natural as possible when using the system, not stilted or ping-pong. (That is, the system should allow the user to "rattle off" a string of items he wants, without waiting for each individual prompt to come from the WPC.)

[001325] Additional benefits Constraining the vocabulary provides several other developmental and functional benefits:

We can use a less expensive, less sophisticated speech recognition system, which means we have more vendors to choose from.

The speech system will consume less RAM, leaving more memory free for other wearable components and systems.

A constrained vocabulary requires less processing power, so speed won't be compromised.

We can use speech recognition engines that are tuned to excel in high-ambient-noise environments.

[001326]      Ideas and implications    The UI benefits from a dynamic vocabulary but also benefits from escape mechanisms to deal with words the engine has trouble recognizing algorithmically, such as foreign words. Thus, it is preferable to constrain grammar and vocabulary or, if unavoidable, to filter it further (e.g., 500 entries in contacts). Should make it clear which part of the UI the user is supposed to interact with, vs. which parts he's only has to read. It should accommodate the linearity of speech.

[001327]      Some important words to recognize: days of the week, months of the year, 1-31, p.m., a.m., currency, system terms such as Page Down, Read, Reply, Forward, Back, Next, Previous, and Page Up. The UI should listen for certain words for itself (system terms), plus ones for the Applet (Applet terms).

#### THE EXAMPLE UI WILL OFFER MULTIPLE WAYS TO SELECT ITEMS BY VOICE.

Supported UI Requirement: 0 The UI should let the user direct the system under any circumstances.

Supported UI Requirement: 0 The UI should help the user manage his privacy.

[001328]      Rationale    Because there will be no speech training in the UI—e.g., no way to correctly pronounce Jim Rzygecki and have the WPC find it in the list—the UI should have an alternative method for accepting items it doesn't recognize. In other circumstances, the system may be able to interpret the name or command word, but the user may want to keep the content of such an interaction private while still using his voice. (For instance, if he's on a subway and doesn't want others to know he's making a stock buy with his financial advisor.)

[001329] Ideas and implications The user might be able to choose the number or letter of an item in a list rather than state the name of the item itself. He might also be able to voice-spell the first few letters of the name.

THE EXAMPLE UI WILL WORK WITH MANY WPC APPLETS AT  
ONCE.

Supported UI Requirement: 0 The UI should work with multiple WPC applications.

Supported UI Requirement: 0 The UI should allow and assist with multitasking and switching among WPC applications.

Supported UI Requirement: 0 The UI should promote the quick capture and deferral of ideas for later processing.

[001330] Rationale A WPC can readily support the multi-tasked, stream-of-consciousness thinking and working methods that most people perform dozens of times a day. By combining Applets and connecting related information across them, a user can streamline his efforts and the WPC can more easily store and call up context-specific data for him.

[001331] Ideas and implications At the very least, the Example UI should support:

- E-mail (MAPI)
- Phone (TAPI)
- Location (GPS)
- Calendar/Appointments/Datebook
- XML Routines
- Forms creation—collect and commit information to a database
- Web linking
- Reading of online manuals
- Camera
- Scanning

Video and voiceover input—to use a radio/video machine—to talk to others and see what I see.

Capture of natural data, scan UPS codes, talk to systems, scrawl down something as pictures, take photos just to capture information.

THE EXAMPLE UI WILL LET THE USER DEFER WORK AND PICK UP WHERE HE LEFT OFF.

Supported UI Requirement: 0 The UI should allow and assist with multitasking and switching among WPC applications.

Supported UI Requirement: 0 The UI should help the user manage and reduce the mental burden of interacting with the WPC.

Supported UI Requirement: 0 The UI should help the user rapidly ground and reground with each use of the WPC.

Supported UI Requirement: 0 The UI should promote the quick capture and deferral of ideas for later processing.

[001332] Rationale The interruptible nature of using a WPC means the user should be able to defer or resume an activity anytime during the day. Examples include the ability to:

Open a new contact and go to new Applet but come back to where he left off in the contact.

Put something on the back burner as-is so that he can return to it later in the same state in which he left it (rather than putting it all away and starting over).

Pull up several Applets at once if a related series of tasks has been interrupted. (I.e., sequencing as stream of consciousness from one Applet to the next pulls up all related info at once—putting all related, cross-Applet info aside temporarily, rather than closing all, filing away, and reopening everything again. A form of regrounding.)

THE EXAMPLE UI SHOULD ADJUST OUTPUT MODES TO THE  
DESIRED LEVEL OF PRIVACY.

Supported UI Requirement: 0 The UI should help the user manage his  
privacy.

[001333]      Rationale    As wearables become more popular, users will become more  
concerned about social appropriateness and accidental or deliberate  
eavesdropping as they use the system. The Example UI should therefore address  
situational constraints that include a user's desired privacy for:

His interaction with the WPC (concealing whether he's using it or not).

His context for using it (concealing whether he's setting a dinner date or  
selling stock).

His WPC content (concealing what he's hearing or seeing through the  
WPC).

His own identity information (concealing personal information or location  
from others who have WPCs or other systems).

[001334]      Ideas and implications    The UI should be able to detect the user's position  
anonymously rather than, say, have a building tell him (and everyone else) where  
he is. If the UI cannot adequately detect the user's need for privacy automatically,  
it should provide a means for the user to input this setting and then adjust its  
output modes accordingly.

THE EXAMPLE UI WILL USE LISTS AS THE PRIMARY UNIFYING UI  
ELEMENT.

Supported UI Requirement: 0 The UI should let the user direct the system  
under any circumstances.

Supported UI Requirement: 0 The UI should help the user manage and  
reduce the mental burden of interacting with the WPC.

Supported UI Requirement: 0 A WPC UI should be extensible to future technologies..

[001335]        Rationale    If a WPC is to be used in all contexts with the least amount of mental effort, it should not have fundamentally different interaction depending on the input mode. What works for hands-on operation should also work for hands-free operation. Because speech is assumed but is inadequate for directing a mouse, the Example UI will therefore map all input devices and modes to operate from a list. This single unifying element will enable the user to perform any function by selecting individual items from groups of items.

[001336]        Using lists provides the following benefits to users :

              Users can select from lists using any input mode available—speech, pointer, keyboard.

              Having one primary input method lets users extrapolate across the system—learn a stick shift, know all stick shifts.

              Lists simplify operation and promote consistency, which reduce cognitive load and accelerate the user's expertise.

              New input modes (e.g., private) and devices (e.g., eye-tracking) can be mapped in without appreciably affecting the interaction or coding.

              We don't have to care what WPC Applet the list is being applied to—the user just always selects from a list.

[001337]        Ideas and implications    The lists are the data items that pop up to select from using the menus.

              THE EXAMPLE UI WILL BE WINDOWS COMPATIBLE.

              Supported UI Requirement: 0 The UI should accept spoken input.

              Supported UI Requirement: 0 The UI should work with multiple WPC applications.



[001338]        Rationale    This will enable us to leverage the advantages of immense PC market and produce a general-platform product that takes advantage of the uniqueness of a WPC. The Example UI will be a shell that runs inside Windows. The user launches Windows, launches the shell, and then navigates the WPC functionality he wants. The Windows task bar is still visible.

[001339]        Ideas and implications    To make the most of standards, the UI should rely on PC hardware standards, especially for peripherals and connectors. Any new standards we create ought to be designed to be consistent with the rest of PC market. We intend to follow the current power curve and never compromise in power or capability.

## OTHER CONSIDERATIONS

### WHY DON'T CURRENT PLATFORMS WORK FOR WPC USE?

#### THEY CAN'T BE AVAILABLE ALL THE TIME

[001340]        Current platforms can only be interacted with sporadically. A desktop system is only available at the desk. A laptop must be removed from a briefcase, and a suitable surface located. WinCE devices and palmtops must be removed from the pocket. The result is that tasks are deferred until the user can dedicate time to interaction with the platform.

[001341]        This prevents the use of information storage and retrieval to be used as pervasive memory and knowledge augmentation. It makes solutions undependable by introducing the opportunity for lost or erroneous information.

[001342]        As a result of this lack of availability, the system cannot gain the user's attention or initiate tasks. This thwarts opportunities to facilitate daily life tasks.

[001343]        Takeaways for the UI    The wearable PC will allow you to be in constant interaction with your computer. Daily life tasks can be dealt with as they occur, eliminating the delay associated with traditional platforms. The system can act as an extension of your self, and an integral part of your daily life.

## THEY OFFER LIMITED FUNCTIONALITY

[001344] Palmtop devices accomplish greater availability by severely compromising system functionality. They are too under-powered to be good general-purpose platforms. Scaled down “partner products” are often used in lieu of the standard tools available on desktop systems, and many hardware peripherals are unavailable. In general, the ability to leverage the advantages of mainstream hardware and software is lost.

[001345] Takeaways for the UI The wearable PC will be, as far as possible, a fully powered personal computer. It will use high end processors, have large amounts of RAM, and run the Windows operating system. As such, it will leverage all of the advantages enjoyed by laptop and desktop computers.

## THEY CAN'T BE USED IN EVERY ENVIRONMENT

[001346] Even if current computing platforms were continuously available, they would be unusable in their current form. Laptops are unusable while walking. Palmtops are unusable while driving. The sounds that a traditional computer emits are inappropriate to a variety of social settings.

[001347] Additionally, current platforms have no sense of context, and cannot modify their behavior appropriately.

[001348] Takeaways for the UI Both the wearable PC and software will be tailored to use in everyday situations. Eyeglass-mounted displays, one-handed keyboards, private listening, and voice interaction will facilitate use in a variety of real life situations.

[001349] The software will also have a sense of context, and modify its behavior appropriately to the situation. A scaling UI will adapt to accommodate the user's cognitive load, providing subtler, less intrusive feedback when the user is more highly engaged.

## THEY ARE PASSIVE RATHER THAN REACTIVE

[001350] Current solutions tend to work as passive tools, reacting to the user's commands during a productivity session. This is a lost opportunity to gain the attention of the user at the appropriate time, and offer assistance that the user has not requested, and may not have been aware of.

[001351] Takeaways for the UI With a wearable PC, the system can gain your attention in order to suggest, remind, notify, and augment your world in appropriate ways. Our mantra is: "How can we make computing power a proactive participant in daily life?"

### UI EXAMPLES

#### PROTOTYPE A

[001352] Description Built solely on a Windows interface. All visual—no voice used.

[001353] What we learned This prototype has problems because Windows is all two-dimensional. It cannot provide voice-based UI and feedback well. All-visual is sub-optimal for a WPC used in a hands-free environment. The result was a poor cousin to Outlook.

#### PROTOTYPE B

[001354] Description Built on voice recognition to control Outlook and Microsoft Agents to be the focal point for interactions and to handle the voice recognition. The Agents use a hierarchical menu system (HMS). Could try an all-voice, natural language interaction for no-hands use. This prototype integrated with Outlook for contacts, appointments, and e-mail; allowed the user to capture reminders as .wav files (*i.e.*, recorded a note and then played it back at a specific time); and included an applet that we created for taking notes.

[001355] What we learned This prototype had problems because:

The HMS buried commands instead of exposing all the commands at once. It was like using a phone system that forces you to listen through all the options

before choosing which one is right, and meanwhile you may have forgotten the option you wanted.

The Agents locked us into a ping-pong question-answer mode that forced you to hear a question, give a response, wait for the next screen and question, and give another response. The computer couldn't advance without you, and you couldn't advance without waiting for the computer. It was unnatural, stilted, boring, and time-consuming.

All the windows consumed a lot of display space.

This solution provided only one method of input—voice—which is not always appropriate for WPC users.

By providing a single point of action—an Agent that talked to them like a person—people wanted it to work with even more natural language, but it wouldn't. The closer it was to freeform and natural language, the more people gave it ambiguous language and treated it like a real person.

[001356] Takeaways for the UI This prototype influenced several UI decisions:

The goal is to interact with and talk to the WPC just as you would talk to a person taking an appointment. However, the tool should not use 100% natural language—it is too complicated to train the system to each user's style and vocabulary. Voice can be used if the vocabulary is constrained and the user is aware at all times of which words he's allowed to say to get a job done. A semi-formal grammar can constrain the options to specific natural-language vocabulary but still cue the user about his options. It enables the WPC to meet the user halfway.

The tool should provide an environment that's not ping-pong—it should let thoughts flow naturally from one part of a task to the next. A better solution would be to let the customer rattle off all the attributes desired (such as make an

appointment with Bob for Tues June 13 at 12:30 and O'Malley's). Preferably, the system would let you say those things in any order.

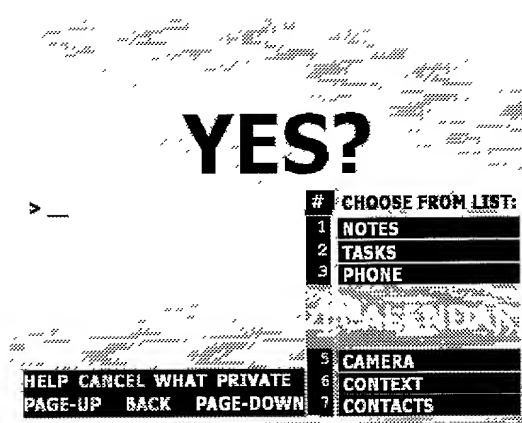
The tool should provide alternatives to voice input at the same time that it provides voice input—voice alone is sub-optimal because it typically involves memorization and privacy to interact. Also, all-voice doesn't expose all the commands and options very well.

Agent technology is a poor UI choice for a WPC UI. It is bolted on to a system, rather than integral to it, and inflexible in how it can be used. In addition, its anthropomorphic nature caused people to try to interact inappropriately with the WPC.

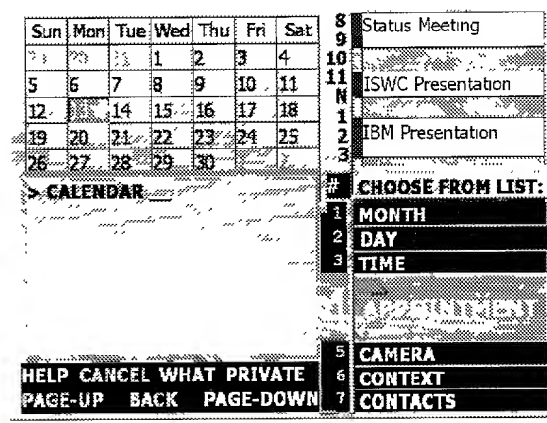
#### PROTOTYPE(S) C

[001357] Description The many flavors of this version seek to blend voice, audio, and hands-on use. It uses a constrained voice recognition vocabulary and presents choices along the bottom that are specific to each Applet. (This row of choices has been referred to as the "bouncing ball." It represents the steps the user goes through to complete any task. For instance, in the Calendar Applet, the steps for making an appointment might be Who, When, Where, What.) The choices are "meta commands" that are always present and, when selected, lead to lists that show the choices available for each step of the bouncing ball. The vocabulary can cross over to other applets using the same verbs or tasks. The words you can say are all in gold. The UI offers both audio and visual prompts to guide the user from one step to the next.

[001358]

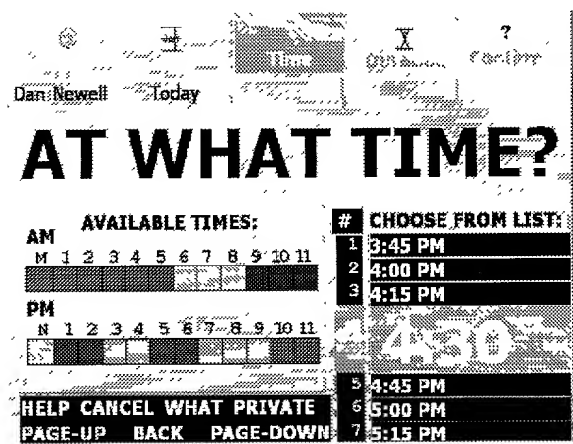


Starting the Calendar (early rendition).

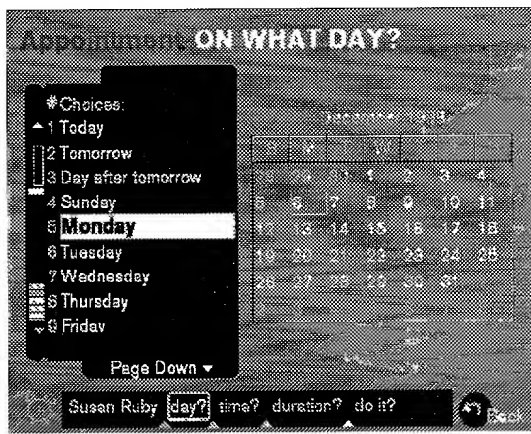


Starting an appointment (early rendition).

[001359]



Completing one of the bouncing ball steps



Example UI

[001360]

What we learned There are several elements that work well about this UI:

The consistent order of the bouncing-ball choices defines a pattern that you can learn and follow to speed up interaction. It helps you learn “the idiom”—the correct order for rattling off information at natural speaking speed so the computer can follow it. It also allows a semi-formal grammar to be imposed while still supporting voice recognition.

The bouncing ball lets you see the options before you navigate with the voice—you know what the holes are that can be filled when using the Applet.

The bouncing ball choices can be either clicked like a button or spoken, supporting both hands-free and hands-on use.

The gold text visually alerts you to what can be said. If you can't see it, you can't say it.

The who/what/where/when construction is always available—you never get a blank slate.

What you do is simple:

See the choices.

Make a choice.

Get a new set of choices.

If you want to know what you can do, look at the list, the bottom bar, or the gold text.

You only learn one input method, and it always works the same, no matter what list you're using.

The goal is to get the user to adapt to the system and to have the system meet them halfway. An all-natural-language solution would have the system totally adapting to the person.

#### UI METHODS SUPPLEMENTING OTHER IDEAS

[001361] Learning Model – attributes that characterize the preferred learning style of the user. The UI can be changed over time as the different attributes are used to model to optimal presentation and interaction modes for the UI, including user preference.

[001362] Familiarity – a simpler model than Learning, in part, it focuses on characterizing a user's learning stage. In the designs shown, there is duplication in UI information (e.g. the prompt is large at the top of the box, implicitly

duplicated in the list of choices, and it also appears in the sequence of steps in the box at the bottom of the screen). As a user becomes more familiar with a procedure, the duplication can be eliminated.

[001363]        User Expertise – different from Familiarity, Expertise models a user's competence with their computing environment. This includes the use of the physical components of the system, and their competence with the software environment.

[001364]        Tasks – characteristics of tasks include: complexity, seriality/parallel (e.g. you may want the system to provide the current time at any random moment, but you would not being able to use the command "Repeat" without following a multi-step procedure.), association, thread, user familiarity, security, ownership, categorization, type and quantity of attention for various use modes, use, prioritization (e.g. urgent safety override), and other attributes allowing the modeling of arbitrarily complex models of a task.

[001365]        Reasons to Scale:

[001366]        Urgency – especially of data

[001367]        Collaboration – with other's, especially if they are interacting via their computer

[001368]        Security – not the same a privacy, this is weather the user and data match minimum security levels

### PROMINENCE

[001369]        Prominence is the relative conspicuousness of a UI element(s). It is typically achieved through contrast with other UI elements and/or change in presentation.

### USES

[001370]        Communicate Urgency

[001371]        Communicate Importance



- [001372] Reduce acquisition / grounding time
- [001373] Reduce cognitive load
- [001374] Create simplicity
- [001375] Create effectiveness

## IMPLEMENTATION

### AUDIO

- [001376] Volume, Directionality (towards front of user), Proximity, tone, 'irritable' sounds (*i.e.* fingernails across a chalkboard), and changes in these properties.

### VIDEO

- [001377] Size, intensity of color, luminosity, motion, selected video device (some have greater affinity for prominence), transparency, and changes in these properties.

### HAPTIC

- [001378] Pressure, area, location on body, frequency, and changes in these properties.

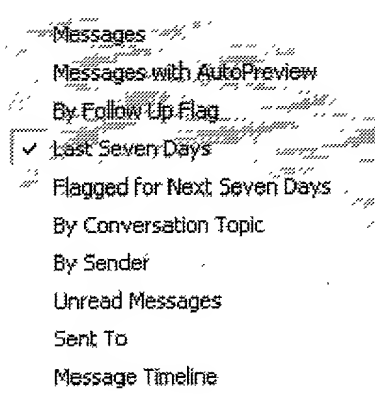
### PRESENTATION TYPE

- [001379] Haptic vs. Audio vs. Video
- [001380] Multiple types (associating audio with video; or Haptic with audio, etc.)

### ORDER OF PRESENTATION

- [001381] For example, putting the most commonly needed information towards the beginning of a process.

### ASSOCIATION



[001382] Some examples of relationships are common goal (all file operations appearing under a file menu), hierarchy, function, etc.

#### USES

##### CONVEY SOURCE OR OWNERSHIP

[001383] Reduce acquisition / grounding time

[001384] Reduce cognitive load

[001385] Create simplicity

[001386] Create effectiveness

#### IMPLEMENTATION

[001387] Similar presentation (same methods as Prominence)

[001388] Proximity of layout

[001389] Contained within a commonly bounded region. *E.g.* group boxes and windows

#### INVITATION

[001390] Creating a sense of enticement or allurement to engage in interaction with a UI element(s). Beginning of Exploration. "Impulse Interaction"

#### USES

[001391] Create Learnability (through explorability)

#### IMPLEMENTATION

[001392] Explicit suggestion

- [001393] Safety (non-destructive, reversible)
- [001394] Safety (not get lost)
- [001395] Familiarity
- [001396] Novel/New/Different
- [001397] Uniqueness (if all familiar & one new; choose new, if all strange & one familiar; choose old)
- [001398] Quick/Cheap/Instant Gratification
- [001399] Simplicity of Understanding
- [001400] Ease of Acquisition and Invocation/Prominence
- [001401] Rest/Relaxation
- [001402] Wanted/Solicited/Applause
- [001403] Curiosity/Glimpse/Preview
- [001404] Entertainment
- [001405] Esthetics/Shiny/Bright/Colorful
- [001406] Promises: titillation, macabre, health, money, self-improvement, knowledge, status, control
- [001407] Stimulating (multiple sense), increased rate of change
- [001408] Fear avoidance

#### SAFETY

- [001409] A computer can enhance the safety of the user by either providing or emphasizing information that identifies real or potential danger or suggests a course of action that would allow the user to avoid danger, or a computer can suppress the presentation of information that may distract the user from safe actions, or it can offer modes of interaction that avoid either distraction or actual physical danger. An example of the latter case is when the physical configuration of the computer itself constitutes a hazard, such as having the physical burden of

peripheral devices like keyboards which occupy the hands and offer opportunity for the device to strike or become entangled with the user or environment.

### USES

- [001410] Help create learnability
- [001411] Help create effectiveness

### IMPLEMENTATION

- [001412] The implication that interaction will not result in unintended or negative consequences. This can be created by:
  - [001413] Reversibility
  - [001414] Clarity/Orientation cues
  - [001415] Familiarity (not unknown)
  - [001416] Metaphor (Which button is safer? Juggling chainsaws, Grandma w/tray of cookies)
  - [001417] Consistent Mental Model
  - [001418] Full disclosure
  - [001419] Guardian (stop me before I do something dangerous: intervention)
  - [001420] Advisor (if I get confused, easy to get unconfused: solicitation)
  - [001421] Expert Companion (helps me make good decision)
  - [001422] Trusted Companionship (could be golden lab)

### METAPHOR

- [001423] A UI element(s), with a presentation that is evocative of a real world object, implying an obvious interaction and/or function (provides “meaning”).

### USES:

- [001424] Create Learnability
- [001425] Create Simplicity
- [001426] Create Effectiveness
- [001427] Reduce cognitive load

[001428] Reduce acquisition / grounding time

#### IMPLEMENTATION

[001429] Examples: Recycle Bin

#### SENSORY ANALOGY

[001430] Expressing (by design) a UI Building Block(s)' presentation and or interaction with a sensory experience, in order to bypass cognition (work within the pre-attentive state) and take advantage of innate sensory understanding.

[001431] Mouse / Cursor interaction.

#### USES

[001432] Reduce cognitive load

[001433] Reduce acquisition / grounding time

[001434] Create simplicity

[001435] Create effectiveness

[001436] Help create learnability

#### IMPLEMENTATION

[001437] Example: Conveying the location of a nearby object by producing a buzz or tone in 3D audio corresponding to the location of the object.

#### BACKGROUND AWARENESS

[001438] A Sensory Analogy with low Prominence.

[001439] A non-focus output stimulus that allows the user to monitor information without devoting significant attention or cognition. The stimulus retreats to the subconscious, but the user is consciously aware of an abrupt change in the stimulus.

#### USES

[001440] Reduce cognitive load

[001441] Reduce acquisition / grounding time

[001442] Create simplicity

[001443] Create effectiveness

[001444] Help create learnability

#### IMPLEMENTATION

[001445] Example: Using the sound of running water to communicate network activity. (Dribble to roaring waterfall)

#### REASONS TO SCALE

##### PLATFORM SCALING

##### POWER SUPPLY

[001446] We might suggest the elimination of video presentations to extend weak battery life.

##### INPUT / OUTPUT SCALING

##### PRESENTATION REAL ESTATE

[001447] Different presentation technologies typically have different maximum usable information densities.

[001448] Visual – from desktop monitor, to dashboard, to hand-held, to head mounted

[001449] Audio – perhaps headphones support maximum number of distinct audio channels (many positions, large dynamic range of volume and pitch)

[001450] Haptic – the more transducers, the more skin covered, the more resolution for presentation of information.

##### USER ADAPTIVE SCALING

##### ATTENTION / COGNITIVE SCALING

[001451] Use Sensory Analogy

[001452] Use Background Awareness

[001453] Allow user option to “escape” from WPC interaction

[001454] Communicate task time, urgency, priority

## PRIVACY SCALING

- [001455] Use of Safety
- [001456] H/W 'Affinity' for Privacy

## PHYSICAL EMBURDENMENT SCALING

- I/O Device selection (hands free vs. hands)
- Redundant controls
- Allow user option to "escape" from WPC interaction
- Communicate task time, urgency, priority

## EXPERTISE SCALING

- [001457] Scaling on user expertise (novice to expert). Use of shortcuts / post processing.

## IMPLEMENTATIONS

- [001458] These are examples of specific UI implementations.

## ACKNOWLEDGEMENT

- [001459] Constrain to a single phoneme (for binary input)  
L/R eye close, hand pinch interactions

## CONFIRMATION

- [001460] Constrain to a single phoneme (for binary input)  
L/R eye close, hand pinch interactions

## LISTS

- [001461] For choices in a list:
- [001462] Many elements: characterize with examples
- [001463] Few elements: enumerate

## WINDOWS LOGON ON A WEARABLE PC

### TECHNICAL DETAILS

[001464] Winlogon is a component of Windows that provides interactive logon support. Winlogon is designed around an interactive logon model that consists of three components: the Winlogon executable, a Graphical Identification and Authentication dynamic-link library (DLL)—referred to as the GINA—and any number of network providers.

[001465] The GINA is a replaceable DLL component that is loaded by the Winlogon executable. The GINA implements the authentication policy of the interactive logon model (including the user interface), and is expected to perform all identification and authentication user interactions. For example, replacement GINA DLLs can implement smart card, retinal-scan, or other authentication mechanisms in place of the standard Windows user name and password authentication.

### THE PROBLEM TO BE SOLVED

[001466] The problem falls into three parts:

[001467] Provide a paradigm Windows logon (logon mechanism consistent with our UI paradigm)

[001468] Allow for private entry of logon information

[001469] Allow for security concerns (ctrl-alt-del)

### BIOMETRICS

[001470] By scanning your fingerprint, hand geometry, face, voice, retinal or iris, biometrics software can quickly identify and authenticate a user logging on to the network. This technology is available today, but requires extra hardware, and thus may not be appropriate for an immediate solution.





[001476] The windows logon mechanism for this is to require the user to press CTRL-ALT-DEL to get to the logon program. If there is a physical keyboard attached to the WPC, this mechanism can still be used. A virtual keyboard (including the Windows On-screen keyboard) cannot be trusted for this purpose. If there is not a physical keyboard, the only other reliable mechanism is for the user to power down the WPC and power it back up (cold boot).

## INTERFACE MODES

### OUTPUT MODES

[001477] The example system supports the following interface output modes:

HMD

Touch screen

Audio (partial support)

[001478] The interface's primary output mode is video, *i.e.*, HMD or touch screen. Although the touch screen interface is fully supported, the interface design is optimized for an HMD. For this release, audio is a secondary output mode. It is not intended as a standalone output mode.

### INPUT MODES

[001479] The example system supports the following interface input modes:

Voice

1D Pointing Device (scroll wheel and two buttons)

2D Pointing Device (trackball with scroll wheel and two buttons)

Touch Screen (with left and right button support)

Physical Keyboard (standard PC keyboard)

Virtual keyboard (provided as part of the example system)

[001480] Although all input modes are fully supported, the interface design is optimized for voice and for 1D pointing devices.

## HYBRID 1D/2D POINTING

[001481] Moving the trackball moves the pointer. List items (and other active screen objects) provide mouse-over feedback (focus) in the form of highlighting.

[001482] Rotating the scroll wheel moves the highlighting bar up and down in the list. The list itself does not move unless the user scrolls past the last visible item, which causes the next item to scroll into view. Rotating the scroll wheel also hides and disables the pointer. The pointer becomes visible and is reactivated as soon as the trackball is moved.

[001483] Single-clicking the left button causes one of the following:

If the pointer is visible and over a valid target (a list item, the System Menu icon, the Back button, Page Up, or Page Down), then the target is selected.

If the pointer is not visible or not over a valid target, then the currently highlighted list item is selected.

[001484] Single-clicking the right button opens the system menu.

[001485] The user can abort selection by moving the pointer off any valid target before releasing the left mouse button.

[001486] The user can disable 2D pointing entirely as a system preference setting.

## INTERFACE DESIGN

### VISUAL DESIGN

#### LAYOUT

[001487] The example system's visual user interface consists of five basic components in a standard layout:

Applet Tag:	Prompt:
Interface Field	Application Display
Task Orientation	

## FONT

- [001495] By default, all text in the example system is displayed using 18-point Akzidenz Grotesk BE Bold.

## COLORS

- [001496] Prompts are white. Speakable screen objects (can be activated using a voice command) are gold. Disabled speakable objects are dark gray/dark gold. All other text is light gray. (Commands that are permanently disabled should be removed from the list.)

## FRAME COMPONENTS

### APPLET TAG

- [001497] Identifies the current applet. The Applet Tag exists in the visual interface only—it has no audio equivalent.

### PROMPT

- [001498] The prompt indicates to the user what s/he should do next. The system speaks the prompt as soon as the screen appears and displays the prompt in the designated area along the top edge of the screen. Users can issue voice commands even while the system is speaking a prompt. As soon as the system recognizes a valid voice command, it stops speaking the prompt and confirms the voice command (unless the user has disabled audio feedback for prompt confirmations, in which case it speaks the next prompt).

- [001499] As a rule, audio and video prompts should use identical wording. Exceptions should be made only if alternative wording has been demonstrated to enhance usability.

## INTERFACE FIELDS

- [001500] Interface fields serve two functions:
- [001501] They reveal to users the range of appropriate responses to the current system prompt.
- [001502] They allow users to communicate their responses to the system.
- [001503] Four types of interaction field are supported by the example system: single selection lists, multiple selection lists, data entry fields, and trees. By default, interface fields are spoken by the system only when the user invokes the “list” command.

## LISTS

- [001504] A list is a set of appropriate user responses to the current prompt. Each response is presented as a numbered item in the list.
- [001505] In lists, the input focus — which indicates where the user’s input is being directed — is shown by highlighting the currently targeted list item. Only one screen object can have the input focus at any time. By default, the first item in a list has the input focus. Selection — which indicates the current value of each list item — is shown by checking the item. Depending on the input device used, input focus and selection may or may not always move in tandem. Depending on whether the list is single or multiple selection, one or more list items may be checked at once. Unless an application specifies otherwise, focus defaults to the first list item.
- [001506] Several types of visual feedback are associated with selection. On mouse-down, the selected menu item becomes checked. On mouse-up, the highlighting blinks.

[001507] Lists can contain more items than can be shown simultaneously. In this case, a scrollbar provides a visual indicator to the user that only a portion of a list is visible on the screen. When the user moves the mouse wheel beyond the last currently visible item, the next item in the list scrolls into view and becomes highlighted. List items move into view in single increments.

[001508] The size of the scroll box represents the proportion of the list content that is currently visible. The position of the scroll box within the scrollbar represents the position of the visible items within the list.

### LIST INTERACTION

[001509] The Example UI supports list interaction through 1D (scroll wheel) and 2D (trackball, touch screen) pointing devices, voice commands, and keyboard.

#### 1D POINTING DEVICES

[001510] When using a scroll wheel as a 1D pointing device, the user moves the input focus by rotating the scroll wheel and makes selections by clicking the left mouse button. With 1D pointing devices, focus and selection are independent: highlighting moves whenever the scroll wheel is rotated, but a checkmark doesn't appear until the left mouse button is clicked.

[001511] When the scroll wheel is rotated, the pointer is hidden and disabled; it remains so until the pointer is moved via the trackball or other 2D input device.

#### TRACKBALL

[001512] When using a trackball as a 2D pointing device, the user moves the input focus by moving the pointer over the list items and makes selections by clicking the left mouse button. As with a scroll wheel, focus and selection are independent. The UI provides mouse-over highlighting for list items, but a checkmark doesn't appear until a selection is made. The user can abort a selection by moving the pointer off a valid target before mouse-up.

## TOUCH SCREEN

- [001513] When using a touch screen as a 2D pointing device, touching a list item moves both the input focus and the selection to the list item; the user cannot move highlighting independently from checking. The user can abort a selection by moving the pointer off a valid target before lifting her finger.

## VOICE COMMANDS

- [001514] When using voice commands, the user selects a list item by speaking it. (See also the section below on coded voice commands.) As with touch screen interaction, input focus and selection always move in tandem. Users can speak a list item that isn't currently visible. In this case, the selected list item is scrolled into view before checking it to give the user visual feedback for selection.

## KEYBOARD

- [001515] When using a keyboard to interact with lists, the user moves the input focus by pressing the up and down arrows and makes selections by pressing the enter key. In this case, focus and selection can be controlled independently.

## SINGLE SELECTION LISTS

- [001516] In single selection lists, selecting one item automatically unselects all other items. The user can invoke the "Next" system command to select the list item that currently has the focus.

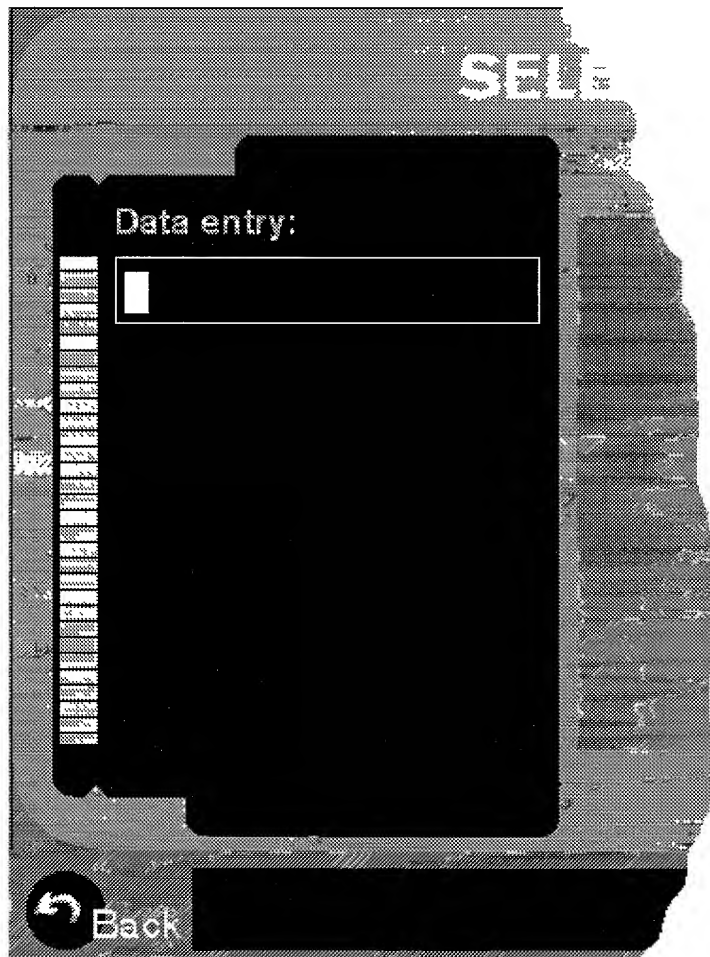
## MULTIPLE SELECTION LISTS

- [001517] In multiple selection lists, selecting an item toggles it between the selected and unselected state. Selecting one item has no effect on the selection status of other list items. With certain input methods (e.g., scroll wheel, keyboard arrows), selection and focus may diverge as the user moves the focus without changing the selection. At the moment a selection is made, the focus shifts to the just-selected item. With other input methods (e.g., 2D pointer, voice), the focus and

selection always move in tandem. The user should invoke the "Next" system command to indicate s/he is finished selecting items in a multiple selection list.

### DATA ENTRY FIELDS

[001518] A data entry field is a container for free-form alphanumeric data entry and editing. It can be defined to support a single line or multiple lines of text. Characters can be entered and edited in a data entry field using a physical or virtual keyboard, voice recognition, or handwriting recognition. Like the other interface fields, data entry fields appear in the central left portion of the frame, as shown below.



[001519]



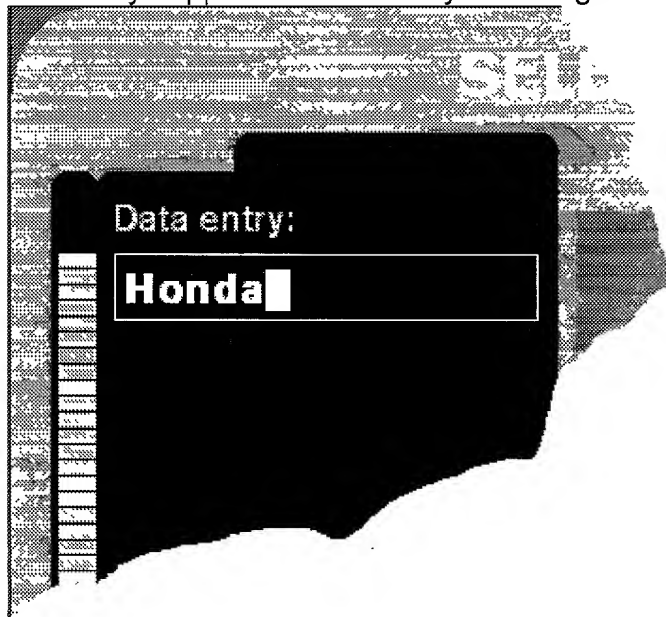
## FOCUS

[001520] To enter or edit text, the keyboard should have the input focus, which is indicated by the presence of a blinking cursor (as shown above). When the keyboard does not currently have the input focus, the input area's outline box and text colors change from white to gray, and the cursor disappears.

[001521] Because interface fields are displayed one at a time, input focus shifts to the keyboard input area automatically (e.g., when a frame with a text entry field opens, or when the user closes the system commands menu). However, keyboard and voice commands can target the input focus to specific characters within the data entry field.

## ENTERING AND EDITING DATA

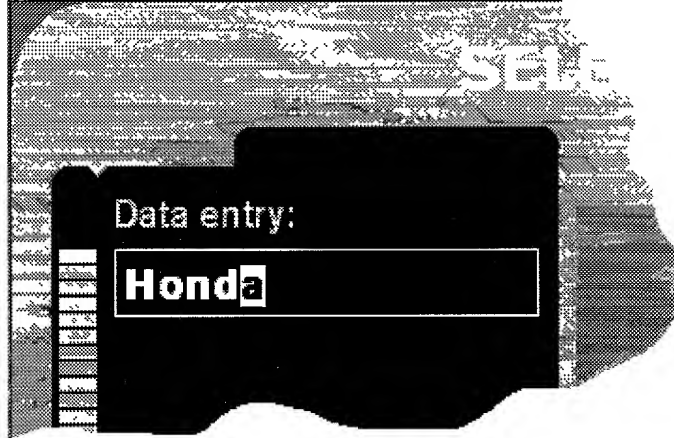
[001522] When entering data into an empty field, characters are inserted at the cursor. As each character is inserted, the cursor moves one space to the right; the cursor always appears immediately to the right of the last inserted character.



[001523]

[001524]

[001525] Editing a data entry field is limited to backspacing and retyping. Backspacing when the cursor is at the end of the data string moves the input focus to the preceding character. When input is focused on a character, the character appears in reverse color within the cursor, as shown below.



[001526]

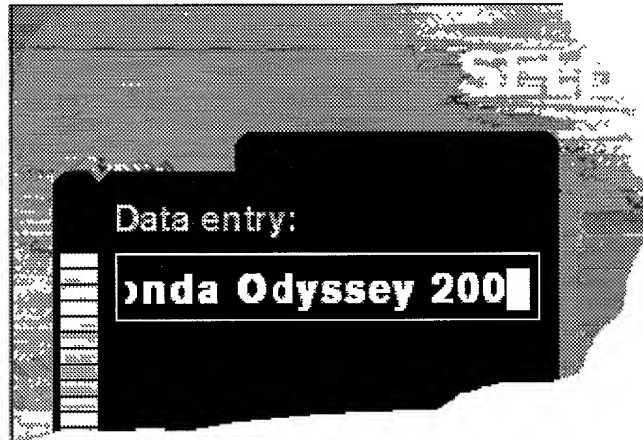
[001527] Backspacing when the input focus is already over a character deletes that character and again moves the cursor back to the preceding character. Once the incorrect characters have been removed, the user can type the correct characters.

[001528] By default, the system provides only visual feedback as each character is typed. As an option, however, the user can invoke an echo mode, in which the system speaks each character as it is typed. The user can toggle echo mode on and off by pressing the "Echo" key on the virtual keyboard or by enabling echo feedback in the system preference settings.

#### MAXIMUM LENGTH

[001529] A maximum length should be specified for every data entry field, although the maximum length may be greater than the field can display simultaneously. For example, a data entry field may have a maximum length of 30 characters, even if only 15 can be displayed at once. If the user types in text that is too long to display in a data entry field, then the text scrolls to allow the user to see each

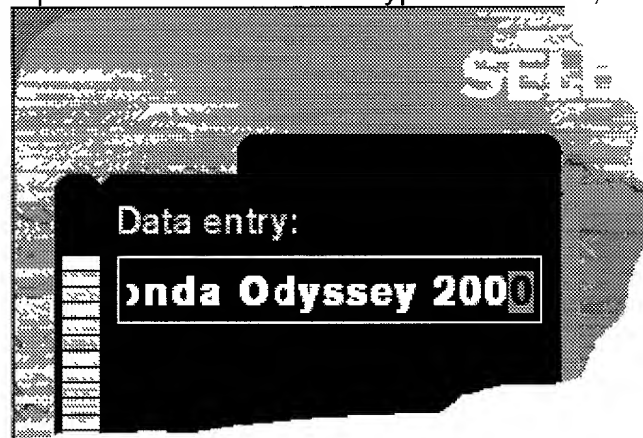
character as it is entered. In a single-line data entry field, the text scrolls to the left, as shown below.



[001530]

[001531] In a multiple-line data entry field, the text scrolls up. To reveal text that has scrolled out of view, the user moves the cursor to the desired text. (There are no scroll bars on data entry fields.)

[001532] When the user reaches the data entry field's maximum length, the cursor changes from white to red; instead of advancing to the next space, the cursor remains positioned over the last-typed character, as shown below.



[001533]

[001534] If the user continues typing, the last character in the field is overwritten and an error sound is played. If the user overtypes the last character in a field three

times in a row, then an error message appears, explaining the maximum length for the data entry field.





## SUBMITTING AND ABORTING

[001535] When data has been entered to the user's satisfaction, s/he issues a voice or keyboard "Enter" command to submit the data. The data is saved to the field and the next frame is presented to the user.

[001536] If the user wishes to abort data entry (i.e., discard any changes made to the data entry field), s/he issues a "Cancel" (voice or mouse) or "Escape" command (virtual or physical keyboard).

[001537] The table below summarizes data entry field interactions with the supported input methods.

**Table X. Data entry field interactions**

Interaction	Input Device			
	Virtual Keyboard	Speech Recognition	Handwriting Recognition*	Physical Keyboard
<b>Enter a character</b>	Select the character	Say the character	Draw the character	Press the character key
<b>Delete a character</b>	Select the "Backspace" key	Say "Backspace"		Press the "Backspace" key
<b>Insert a space</b>	Select the spacebar	Say "Spacebar"		Press the spacebar
<b>Shift (uppercase)</b>	Select the "Shift" key	Say "Uppercase"		Press the "Shift" key
<b>Submit data entry</b>	Select the "Enter" key	Say "Enter"		Press the "Enter" key
<b>Abort data entry</b>	Select the "Escape" key	Say "Cancel"	?	Press the "Esc" key

\* These gestures are recommended. The handwriting recognition engine will ultimately determine the gestures

## INTERACTION DETAILS FOR SPECIFIC INPUT

### METHODS

#### VIRTUAL KEYBOARD

[001538] Users can invoke the virtual keyboard using the "Keyboard" system command. This causes the virtual keyboard to appear on the screen, as shown

below. The pointer changes from an arrow to a hand. As long as the virtual keyboard has the focus, user input is limited to keys on the keyboard. (Plus some non-virtual keyboard way of escaping from the virtual keyboard.) Other interface items and other modes of input are disabled.



[001539]

### SPEECH RECOGNITION

[001540] Users can invoke speech recognition using the "Voice entry" system command. This causes a "speech keyboard" (not yet designed) to appear, providing a list of the voice commands that are available for data entry. The pointer changes to an ear. As long as the speech keyboard has the focus, user input is limited to voice commands on the speech keyboard and valid alphanumeric characters. (Plus some non-speech way of escaping from the speech keyboard.) Other interface items and other modes of input are disabled.

### SPEECH ERROR CORRECTION

[001541] As a supplement to the standard editing methods shown above, two additional methods are provided to help the user correct speech recognition errors.

[001542] The first correction method is to invoke a database of common misinterpretations by saying "Correction." This command, which indicates to the computer that a correction is needed, causes the system to consult the database and suggest alternatives. The system continues to suggest alternatives until the correct character is displayed or the database alternatives have been exhausted.

For example, imagine that the user says, "three," which the system misinterprets as "e." The database might indicate that "e" is a common misinterpretation of "g" and "3." When the user says, "correction," the system replaces the "e" with "g." Since this is still incorrect, the user says, "correction," again. This time, the system correctly replaces the "g" with "3." The error is resolved.

[001543] In the event that the database does not contain the correct character, the user can invoke the third correction method. In this case the system treats the characters as voice-scrollable list. The user can scroll backward and forward through this list using voice commands ("previous character" and "next character") until the correct character is displayed.

For this example, imagine that the user says, "d," which the system misinterprets repeatedly as "z." The user says, "delete," which causes the "z" to disappear. Then s/he says, "e," and the system displays "e." Finally, s/he says, "previous character," and the system replaces the "e" with a "d." Alternatively, s/he could have scrolled back forward from "c" by saying, "next character."

## HANDWRITING RECOGNITION

### TECHNICAL RECOMMENDATION

[001544] The product PenOffice by ParaGraph (<http://www.paragraph.com>) or the Calligrapher SDK by the same company, are possible technologies for implementation.

## INTERFACE

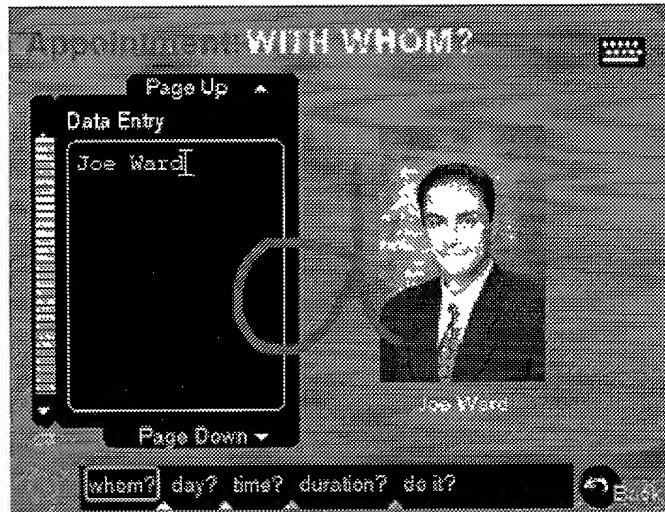
- [001545] Users can invoke speech recognition by using the “Handwriting” system command. This causes a “handwriting palette” (not yet designed) to appear, providing a list of the gestures that are available for data entry. The pointer changes to a hand with a pen. As long as the handwriting palette has the focus, user input is limited to commands on the handwriting palette. (Plus some non-handwriting way of escaping from the palette.) Other interface items and other modes of input are disabled.

## RECOGNITION STYLE

- [001546] Recognition will be on a character-by-character basis, utilizing the entire screen area. The recognition will be style independent, recognizing natural letter shapes, and not requiring any new letter writing patterns (in contrast to Palm’s Graphitti method). The recognition will be writing style independent, recognizing characters that are drawn as cursive, or print, including variations that occur in modern handwriting, like “all caps”, or “small caps”.

## DRAWING THE CHARACTER

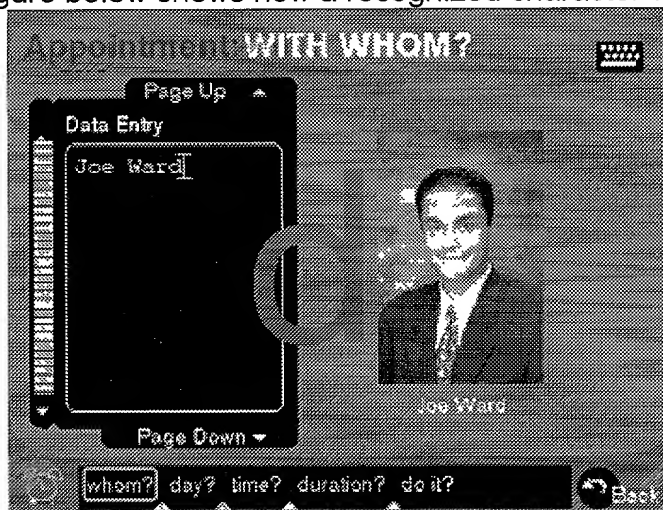
- [001547] The user will be able to “draw” a character on the entire screen surface, with any appropriate 2D input modality. Note that a GlidePoint® could permit finger spelling.
- [001548] The “digital ink” of the characters drawn by the user will be displayed in real time, in a high contrast color, not otherwise reserved for the UI. The figure below shows how a character might appear while being drawn.



[001549]

#### THE RECOGNIZED CHARACTER

[001550] Once the character is recognized, the system will briefly re-display the character as text (e.g., in 128 point font size), centered on the screen and in the same color. The system will also echo back the name of the character, audibly. The figure below shows how a recognized character would be displayed.



[001551]



Enter the gesture for “Enter” to complete the entry

Cancel the input from the system command menu or equivalent.

Select the next field, from the system command menu or equivalent.

## PHYSICAL KEYBOARD

[001553] Users can use a physical keyboard to enter characters into data entry fields simply by typing on the keyboard. Visual feedback is limited to the appearance of the typed characters in the data entry field.


## DATA VALIDATION






[001554] The example system supports within-field and cross-field data validation for text entry fields. When a validation error occurs, an error message appears, explaining the problem and recommending a solution.

## MASKING

[001555] The example system will support masking in data entry fields. Some masks are associated with a unique presentation style to help users enter data in the required format. The following table lists the masks supported for data entry fields and shows the presentation style associated with each mask.

**Table XXX. Masking and presentation styles**

Mask	Presentation Style
Date	 Date

Time	 Time
Date/Time	 Date/Time
Phone	 Phone
Currency	 Currency
Alpha, Numeric, Alphanumeric	 Standard

[001556] In all but the standard presentation style, underscores serve as placeholders for the characters to be entered. As the user enters data, each character replaces its placeholder. For example, if the user types a "1" and a "0" into the currency field shown above, those characters replace the first two underscores:



[001557]

[001558] If a user tries to enter data that conflicts with a data entry field's mask, the character is not displayed; instead, the system plays an error sound. If the user types three illegal characters in a row, an error message appears, explaining the legal entries for the data field.

## TREES

[001559] A command tree is a special type of single selection list that allows commands to be organized and displayed to the user hierarchically. Indentation is used to distinguish the different levels of the hierarchy, which can extend as many as four levels.

[001560] The primary purpose of the tree is to provide “table of contents” navigation for online documentation, but it can be used wherever the user would benefit from viewing commands in a hierarchical structure (e.g., users organized into groups).

[001561] A tree includes two object types: nodes and leaves. Nodes represent branches of the tree and act as containers for leaves, other nodes, or both. Nodes are never “empty.” Leaves represent the lowest level of a branch, and consist of commands or data entry fields. Leaves are never containers. When a tree is used to make a table of contents, the leaf commands are hypertext links to the documentation.

[001562] Selecting a closed node causes that branch to expand, revealing the next level of commands, which could include either nodes or leaves or both. Selecting an open node causes that branch to collapse, hiding all lower levels. Expanding and collapsing an individual node does not affect the state of any other node, so node state is “sticky.”

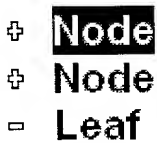
[001563] The user selects a node or leaf by clicking it or speaking it. When a mouse wheel (or other 1D pointing device) is used for navigating a tree, highlighting moves from one item to the next regardless of their relative levels in the hierarchy.

[001564] Each item in a tree consists of an icon and a text string. Three icons should be provided for each tree: collapsed node, expanded node, and leaf. Two icon sets will be included in the SDK: “generic tree” and “table of contents.”

[001565] As an optional feature, nodes and leaves in a tree can be color-coded (or an additional icon?) to reveal whether they contain incomplete data entry fields. (This feature is linked to data validation.)

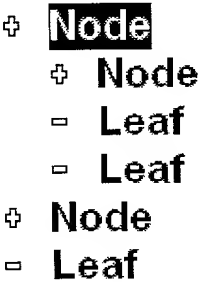
[001566] Another optional feature is to color code leaves that have been visited. This feature is intended primarily for trees used as tables of contents.

[001567] Below is a tree showing only top-level items:



[001568]

[001569] Clicking the highlighted node reveals the next level below that node.



[001570]

### TASK ORIENTATION (BOUNCING BALL)

[001571] The task orientation area provides navigational context to assist in orientation. It is not interactive. The behavior of the task orientation area depends on the current navigational structure.

### LINEAR

[001572] When the user is in a linear navigation structure (*i.e.*, a fixed sequence of frames with no branching, AKA “island of linearity”), the task orientation area displays from left to right the following items:

[001573] The selection made in the previous frame of the linear sequence (if any—not available when the user is still in the first frame of a sequence)

- [001574] The prompt for the current frame (highlighted)
- [001575] Prompts for upcoming frames (as many as will fit on the screen)

### NON-LINEAR

- [001576] When the user is in a non-linear navigation structure, the task orientation area displays from left to right the following items:
  - [001577] The selections made in previous frames (as many as will fit on the screen)
  - [001578] The prompt for the current frame (highlighted)
  - [001579] The user can hide/unhide as a preference setting. If hidden then the screen real estate is available for list and app area
- [001580] Note that the transition may be jarring to the user, and some sort of smooth scrolling transition may be preferable. Further feedback to the user to indicate that they are (or are not) now in a linear process may also be preferable.

### APPLICATION AREA

#### HYPERTEXT NAVIGATION

- [001581] The example system supports hypertext navigation in the application area by 1) converting a hypertext document's links into the items of a list (*i.e.*, a single selection list interface field) and 2) defining a highlight appearance for hypertext links in the application area. When the user scrolls through the list items, the highlighting updates in both the list and in the application area.

#### FULL-SCREEN /PARTIAL-SCREEN DISPLAY

- [001582] By default, the application area occupies only a portion of the total available screen area. However, the user can toggle between partial-screen and full-screen display by using the "minimize" and "maximize" system commands, one or the other of which is always available. These commands are sticky. When the application area is in full-screen mode, all other interface components are hidden except the prompt, which appears in its usual location, superimposed on

the content of the application area. To minimize visual obstruction of the underlying content, the prompt is displayed using a transparent or outline font.

[001583] System commands are available as usual when the application area is in full-screen view. The “system commands” command causes the list of system commands to appear in its usual area, superimposed on the application area, using a transparent or outline font. Although the frame’s interface field is hidden when the application area is in full-screen view, users can still access it through voice or 1D mouse commands. Scrolling the mouse wheel causes the interface field to become visible, superimposed on the content of the application. The interface field disappears again when the user makes a selection or after a brief timeout. If the user makes a selection using a voice command, only the selected item appears.

[001584] When the system is in full-screen view, messages (notifications) will appear and behave as usual, except that they are superimposed over the application content.

## NAVIGATING THE APP AREA

[001585] Users can control what’s visible in the application area by invoking the following commands.

Page up/down (similar to list command)

Scroll up/down/left/right

Zoom in/out

Previous/Next (page)

## SYSTEM COMPONENTS

### SYSTEM COMMANDS

[001586] To reduce recognition errors, system commands are preceded by a universal keyword. By default, the keyword is “computer,” but users can change this keyword as part of the preference settings.

## MENU

- [001587] The “Menu” command causes a list of all system commands to appear in a popup menu. This list appears whenever the user says, “Menu,” clicks the right mouse button, or selects the Menu icon in the upper right corner of the frame. The menu closes as soon as the user issues any system command. (Repeating the “Menu” command closes the menu without performing any other action.)

## QUIT LISTENING/START LISTENING

- [001588] The “quit listening” and “start listening” commands suspend and resume speech recognition. The “quit listening” command is intended primarily for use when ambient noise is misinterpreted by the system as voice commands. Although “quit listening” can be issued as a voice command, “start listening” obviously cannot.

## PREVIOUS/NEXT

- [001589] The “Previous” command navigates to the most recently viewed frame and undoes any action performed as part of the forward frame transition. Data generated by the user in the preceding frame is preserved and displayed to the user. For example, in a tree or single-selection list, the item selected earlier is highlighted; in a multiple-selection list, items selected earlier are checked; in a data entry field, characters entered earlier are present.
- [001590] The “Next” command is enabled only when the user has navigated back one or more frames. This command redoes the action(s) performed the last time the user proceeded through the current frame. The application is responsible for determining when the user can go forward and what data is persisted about the frames that have been backed through. As a guideline, data already entered should be preserved for as long as possible.

## CANCEL

- [001591] The “cancel” command is passed back to the application, which decides how to respond. The intended functionality is to allow users to escape from some well-defined sub-task without saving any input, but it applies to an application-specific chunk of functionality. The command is enabled/disabled by the application on a frame-by-frame basis. When a cancel command is issued, the system displays a warning message, the text for which is supplied by the application, which also determines the button labels and behaviors. We recommend, minimally, allowing the user to proceed with or halt the cancellation. Once the cancellation has been confirmed, the application determines the next state and functionality.

## UNDO

- [001592] The “undo” command reverses the last keystroke-level user action performed within the current frame. It is intended primarily for use with multiple selection lists and data entry fields. If there are no actions within the frame to be undone, this command will be disabled.

## LIST

- [001593] The “list” command causes the system to speak the items currently visible in a list or tree. If the system is currently in number mode, then the system will also speak the item number.

## ONE, TWO, THREE...

- [001594] The number commands allow the user to select list items privately by speaking a number rather than a word. For example, if the second list item happens to “Dan Newell,” the user can say “computer two” and select Dan Newell without revealing the content of the interaction to anyone.



## PAGE UP/PAGE DOWN

- [001595] If the list includes more items than can be displayed simultaneously, the “page up” and “page down” meta-commands can be used to scroll additional items into view.

## EXIT

- [001596] This command returns the user to the startup frame. Application is notified so it can prompt the user to save data.

## NAMESPACE COLLISIONS

- [001597] The following features are intended to allow developers and users manage namespace collisions between system commands and application commands.

The system will expose a standard set of system commands in the UI in two tiers:

Tier1 - require no escape sequence to be accessed: back, cancel, page up, page down.

Tier2 - require an escape sequence to be accessed: system commands, quit listening, list, exit, voice coding.

All system commands can be aliased by the developer or the user as part of the system configuration or by the user at runtime.

The UIF will check at runtime to make sure that there are no namespace collisions between application specific input and the un-escaped system commands. If there is a collision, and the user selects the collided action, the system will prompt the user for disambiguation.

## SYSTEM STATUS

- [001598] Potentially useful information includes: vu meter, battery, speech recognition status, network connectivity.

- [001599] System status - these elements will be part of every frame  
Battery and network signal strength will surface when outside norm (low)

Clock and VU meter will always be on unless user turns them off  
Clock appearance is toggle-able through the configuration settings  
Date/time format is also configurable.

## SYSTEM CONFIGURATION

[001600] User can adjust the following attributes:

### SOUND OUTPUT

[001601] Adjust the volume.

### CLOCK

[001602] Specify whether it is visible and which date/time format to use.

### MICROPHONE

[001603] Launch the setup wizard.

### SPEECH PROFILE

[001604] Switch users.

### SYSTEM COMMAND KEYWORD

[001605] By default, the system command keyword is “computer,” but the user can specify a different keyword.

### SPEECH FEEDBACK

[001606] Several types of speech feedback are available on the example system. Users can enable or disable each type of speech feedback as part of their system preferences.

### ECHO COMMANDS

[001607] When the user selects an item in a list or tree, the system speaks it.

### ECHO CHARACTERS

[001608] As the user enters each character in a data entry field, the system speaks it.

## SPEAK MESSAGES

- [001609] The system speaks the contents of each system message (notification) that appears.

## POINTING

- [001610] The user can disable 2D pointing.

## MESSAGES

### SOURCE

- [001611] The WPC system will manage messages from the following sources:
- Current WPC applet
  - Other WPC applet
  - WPC system
  - OS
- [001612] The WPC system will make no attempt to manage messages from the following sources:
- Non-WPC applications

### MESSAGE TYPES

- [001613] The WPC system should distinguish the following types of message and manage each type appropriately:

[001614]

Message Type	Description	Possible Dismissal Methods
Error	Reports system and application errors to users.	Automatic, Acknowledgement, or Decision
Warning	Warns users about the possible destructive consequences of a user action	Decision (minimally, proceed or cancel)

	and requires confirmation before proceeding.	
Query	Requests task-related information from users before proceeding.	Decision
Notification	Provides information presumed to be of interest to the user but unrelated to the current task.	Automatic, Acknowledgement
Context-appropriate Help	Provides information useful for completing the current task.	Automatic, Acknowledgement

[001633] A unique icon for each type of message will be displayed.

#### PRESENTATION

##### TIMING

##### WITHIN THE USER'S TASK

[001634] Users should be allowed to complete certain tasks (e.g., free-form text entry) without being interrupted by messages unrelated to the current task.

##### WITHIN THE H/C DIALOG

[001635] Messages should be presented by the WPC at a point in the human/computer dialog when the user expects the computer to have the conversational 'token.'

##### ADVANCE WARNING

[001636] The WPC should be able to provide a cue (auditory and/or visual) before presenting any message unrelated to the user's current task.

##### OUTPUT MODES

[001637] The WPC will present all messages in both audio and video.

## MODALITY

[001638] All messages presented by the WPC will be modal. Since the WPC application is itself modal, the effect is that all messages will be system modal.

[001639] If the message is modal, the sound continues until the user responds or (if it is application modal) switches to another application. If the user says something out of bounds or says nothing for a certain period of time, the system repeats the message and prompts explicitly for a response.

## DISMISSAL

### AUTOMATIC DISMISSAL

[001640] The WPC should allow appropriate messages (notification messages and error messages that require no decision from the user) to be dismissed automatically through a timeout.

## USER ACTIONS

### PREEMPTIVE ABORT

[001641] The WPC should allow the user to preemptively abort presentation of a notification message unrelated to the current task. (Requires advance warning.)

### ACKNOWLEDGEMENT

[001642] Users should be able to acknowledge messages using an interaction that is fast and intuitive (e.g., say or click "OK").

## DECISION

[001643] In general, users should be given the opportunity to make a decision any time it would allow them to return immediately to the current task.

## DEFERRAL

[001644] Users should be able to defer rather than dismiss messages when appropriate. Deferred messages should be re-presented automatically after a specified time. Developers should determine whether deferral is appropriate and specify the re-presentation time. (In other words, it is not a requirement that users

be allowed to defer all messages or to specify the re-presentation time for each message.)

## INPUT MODES

[001645] The user should be able to acknowledge, respond to, or defer messages using the following input modes:

Voice

Mouse

Keyboard

Touchscreen

## RE-GROUNDING

[001646] If a message's timing is appropriate (see discussion above), then the WPC will help the user re-ground by presenting the next prompt immediately after the user dismisses a message.

## USER PREFERENCES

[001647] Users should be allowed to

- Turn off the advance warning for messages (if any)
- Specify whether any messages will timeout

[001648] Users should not be allowed to

- Preemptively abort messages that require an acknowledgement or decision

## MODELING BUILDING BLOCKS OF UIS

### SCALING API

### USER/COMPUTER DIALOG MODEL

[001649] This describes a technique for abstracting the functionality of computers in general, and task software in particular, from the methods used to provide the presentation of and interaction with the UI. The functional abstraction is an important part of an ideal system to dynamically scale UI.

[001650]

The abstraction begins with a description of a minimum set of functional components for at least one embodiment of a practical user/computer dialog. As shown in the following Illustration 1, information takes different forms when flowing between a user and their computing environment. The computer generates and collects information with local I/O devices, including many kinds of sensors. The devices provide or receive this information from the computing environment, which may be local or remote. The user perceives computer-generated information, and controls the computing environment with both explicit indications of intent or implicit commands via unconscious gesture or pattern of behavior over time. As long the information is generated by the user or their associated environment and can be detected by the system, it is part of the dialog.

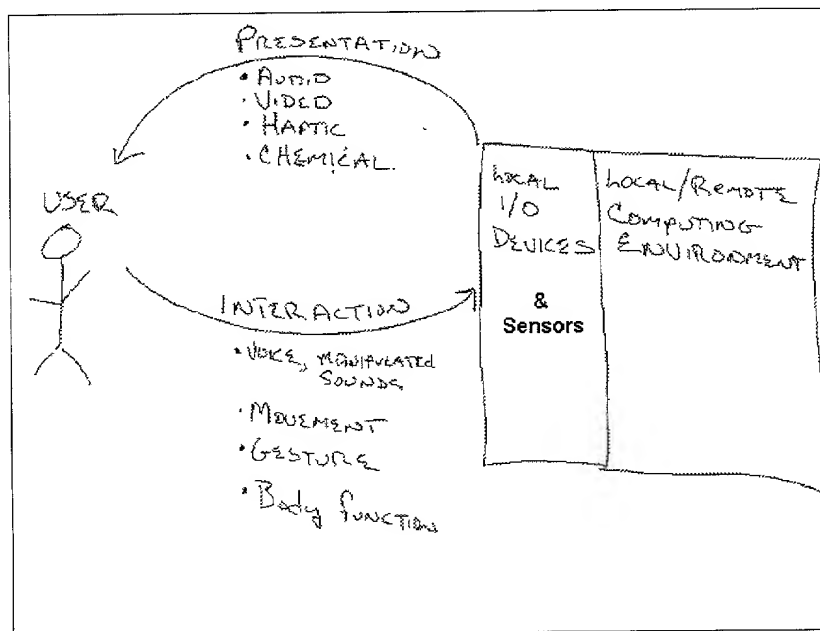


Illustration 1

[001651]

The presentation of information to the user can use any of their senses. This is also true for the user's interaction with the computer's input devices. This

is a significant consideration for the abstraction because it doesn't matter which sense or body activity is being used. In other words, the abstraction supports the presentation and collection of information without regard to the form it is taking.

[001652] In Illustration 2, one embodiment of minimum functional elements are shown.

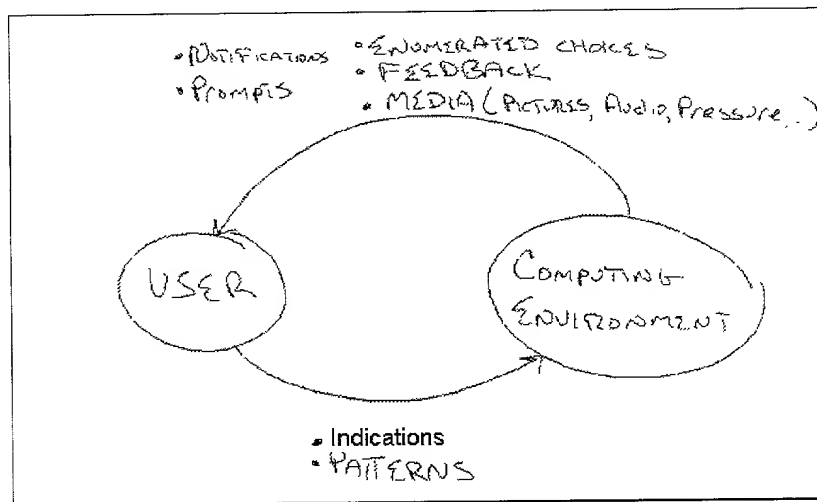


Illustration 2

## COMPUTER TO USER NECESSARY

Prompts – provides user with information regarding an available choice. Types of choices range from unconstrained to constrained. Constrained choices may be enumerated or un-enumerated.

Choice – an option that the user can select which provides information that the computer can act on

Notifications – provides user with information, but does not provide a choice

Feedback – indicates to user what choice has been made



## DESIRABLE

Content – non-interactive

Status – shows progress of system or task related process

Focus

Grouping – relationships between choices

Mode – indication of how system will respond to a choice

## USER TO COMPUTER

### NECESSARY

Indications – these are generated by the user to show their intention.

Intentions are conveyed by selecting choices. Indications do not require a prompt predicate.

### DESIRABLE

Content – this can be any information not designed to indicate a choice to the computer.

Context – Indications and Content that are modeled in the Context Module

Patterns – though not part of explicit user intention, the collection and analysis over time of a user's indications and context can be used to control a computer.

## USER/COMPUTER/TASK DIALOG MODEL

[001653] Since most of the dialog between user and compute relates to the execution of a task, the preceding defining of the important elements of a user/computer dialog is insufficient to completely abstract the task functionality from the presentation and interaction. This is due in part to the desire for the computer system to provide prompts and choices that relate to system control, not to the task.

[001654] Therefore, as shown in Illustration 3, the abstraction can be broken into two pieces:

UI Functionality

Task Functionality

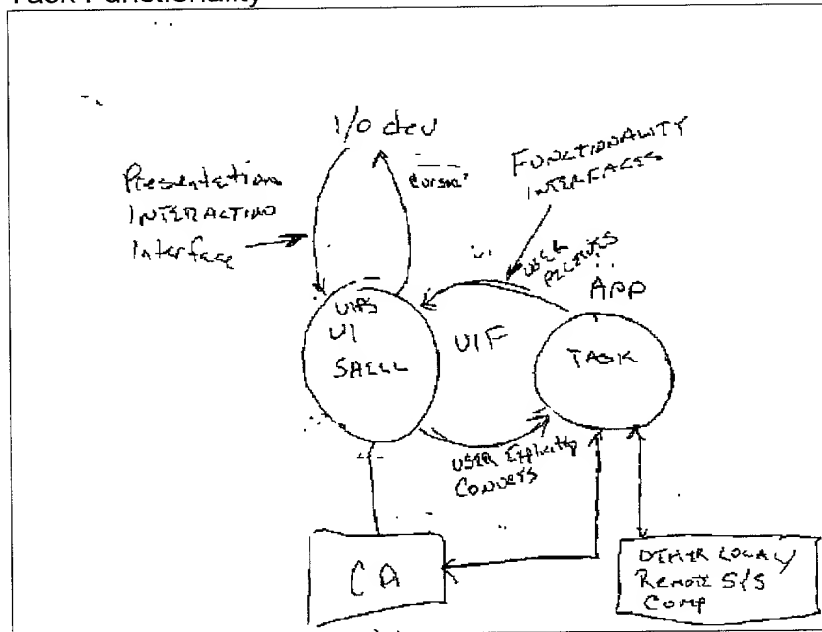


Illustration 3

## UI FUNCTIONS

### INPUT – HOW CHOICES ARE INDICATED

[001655] Devices are manipulated by the user. A computer system could also convert analog signals from devices into digital O/S commands, and interprets them as one of the following:

BIOS or O/S escape sequences

UI Shell commands

### OUTPUT – HOW INFORMATION IS PRESENTED

[001656] Devices, preferences

## TASK FUNCTIONS

### INPUT – WHAT CHOICES ARE INDICATED

Explicit choice indication

Implicit choice indication

## OUTPUT – WHAT CHOICES ARE AVAILABLE

Prompted

Enumerated

Constrained

API: APP→ UIPS:

### 1) ELEMENT OF THE DIALOG

Schema of dialog

get from building blocks

prompts

feedback

### SYNTAX OF DIALOG

<dialog element>

content

<content metadataX>

value

<\content metadataX>

<\dialog element>

### 2) CONTENT OF ELEMENT

may not inform UI changes

text of a prompt

### 3) TASK SEQUENCE/GRAMMAR

How do I string the elements together, navigation path, chunking

[001657]

The following illustration shows chunking on a granular level:

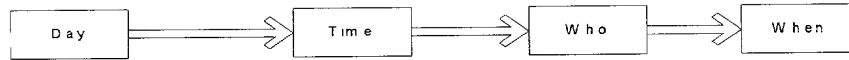


Illustration 4

[001658] If this were a graphical user interface, there would be a separate dialog box or wizard page for each item in the flow chart. In a graphical user interface, chunking on a not-so-granular level is demonstrated by including all these bits of information about creating an appointment in one dialog box or wizard page.

[001659] "navigation state" specifics whether back/next/cancel are appropriate for this step

#### 4) USER REQUIREMENTS WHILE IN-TASK

[001660] This step uses both of the user's hands for the duration of the step, therefore physical burdening = no hands,...

#### 5) CONTENT METADATA

[001661] This is explicit. This data is: sensitive, not urgent, free, from my Mom

[001662] Metadata can include the following attributes:

Sensory mode to user

Characterization of its impact on cognition

Security

To

From

Time

Date

API: APPLICATION ← UIPS

#### 1) & 2) CHOICES WITHIN THE TASK

Value + application prompt

### 3) CHOICES ABOUT THE TASK

Value + system prompt

Back, cancel, next, help, exit,

API: UIPS  $\leftrightarrow$  CA

API: UIPS  $\leftrightarrow$  UI TEMPLATES

API: UIPS  $\leftrightarrow$  CUSTOM RUN TIME UI

API: UIPS  $\leftrightarrow$  I/O DEVICES

### Mechanisms that Dynamically Change a UI

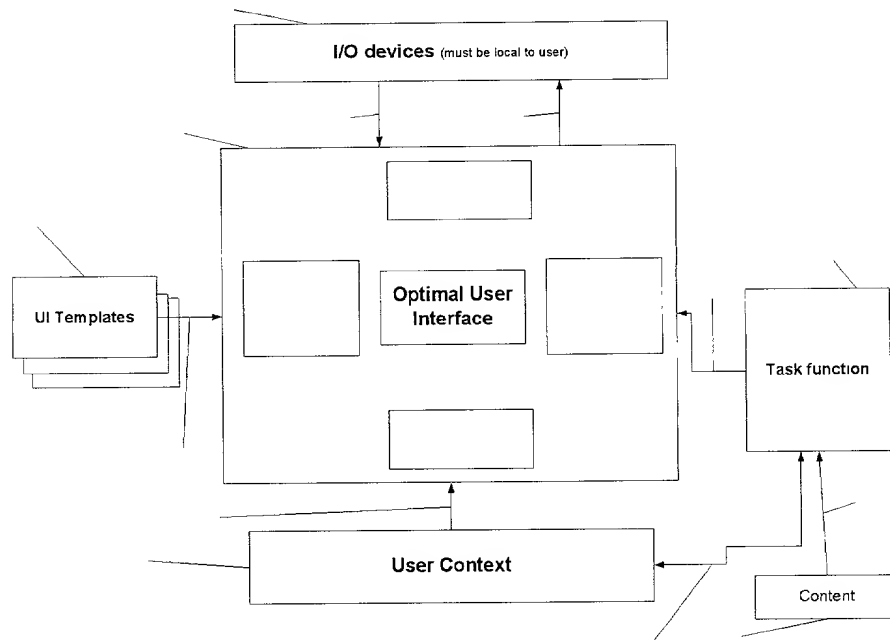


Illustration 5

### OVERVIEW

[001663]

An arbitrary computer UI can be characterized in the following manner.

What are the UI Building Blocks? – What are the fundamental functions of a computer's UI? The fundamental functions are at a very elemental level, such

as prompts, choices, and feedback. A UI element as simple as a command button is a combination of several of these elemental functions, in that it is a prompt (the text label), a command (that is executed when the button is “pressed”), and also provides feedback (the button appears to “depress”).

How are Building Blocks grouped? – What functional structures are created from the building blocks? In Windows these would include dialog boxes, applications, and operating environments, in addition to the basic controls in Windows themselves (scroll bars, command buttons, etc.).

What are General UI Attributes – Ignoring functionality, what are the Gestalt characteristics of a well-designed UI? Some of these attributes include Learnability, Simplicity, Flexibility, and so forth.

#### WHAT ARE THE UI BUILDING BLOCKS?

##### ELEMENTAL FUNCTIONALITY (BUILDING BLOCKS)

[001664] In this embodiment, there are only a limited number of types of user/computer interactions:

User Acknowledgement – User is given a single choice for communication with the computer. *E.g.* clicking okay to acknowledge and error.

User Choices – What is meant here is the *expression* of a choice (that occurs in the user’s mind) to the computer. Moving a cursor, typing a letter, or speaking into a microphone are manifestations of this expression.

PC Notifications – Information presented to the user that is not associated with a choice, such as status reporting.

PC Prompts – The presentation of choice(s) to the user. A command button, by its use of metaphor to imply an obvious interaction, presents a choice to the user (you can click me), and is therefore a kind of prompt.

PC Feedback – presents indications on choices the user has made, or is currently making. When the user clicks on a command button, and the button

appears to become “depressed”, the button is providing feedback to the user on their choice.

## USER CHOICES

[001665] Definition: The user indicates a preference from among a set of alternatives

[001666] WIMP Examples: Choice mechanisms can be ordered by how many choices are available. Low to High:

Confirmations

Lists

Commands

Hierarchies

Data Entry

[001667] Hidden elements can be revealed in various ways. Examples include:

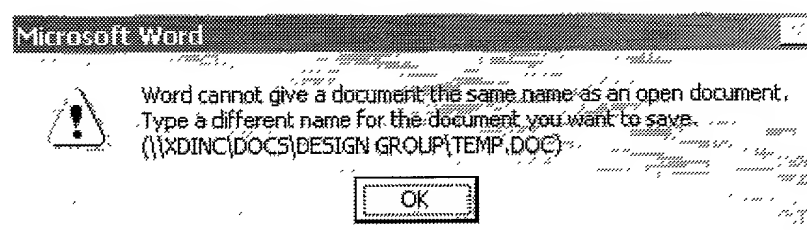
Scroll bar

Text to speech

## ACKNOWLEDGEMENT

[001668] Definition: INFORMING: The PC is alerting the user that it cannot complete an action, and requiring the user to acknowledge that they have received the alert (in contrast to Confirmation below, user has no choices.)

[001669] WIMP Examples:



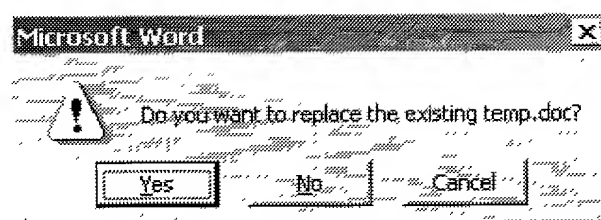
Associated Verbs	Deficits w/WIMP	Alternatives
Acknowledge	Requires either tactile control or speech recognition of name of control (e.g. "OK")	Single choice can be mapped to any utterance. /e.e., blowing on the Example: Using the microphone could suffice
Ignore	Usually UI is stuck in modality	Time out w/ reviewable history

### CONFIRMATION

[001670] Definition: **SEEKING FEEDBACK:** The PC is seeking permission from the user to complete an action that can be accomplished in more than one way, and allows a choice between the alternatives.

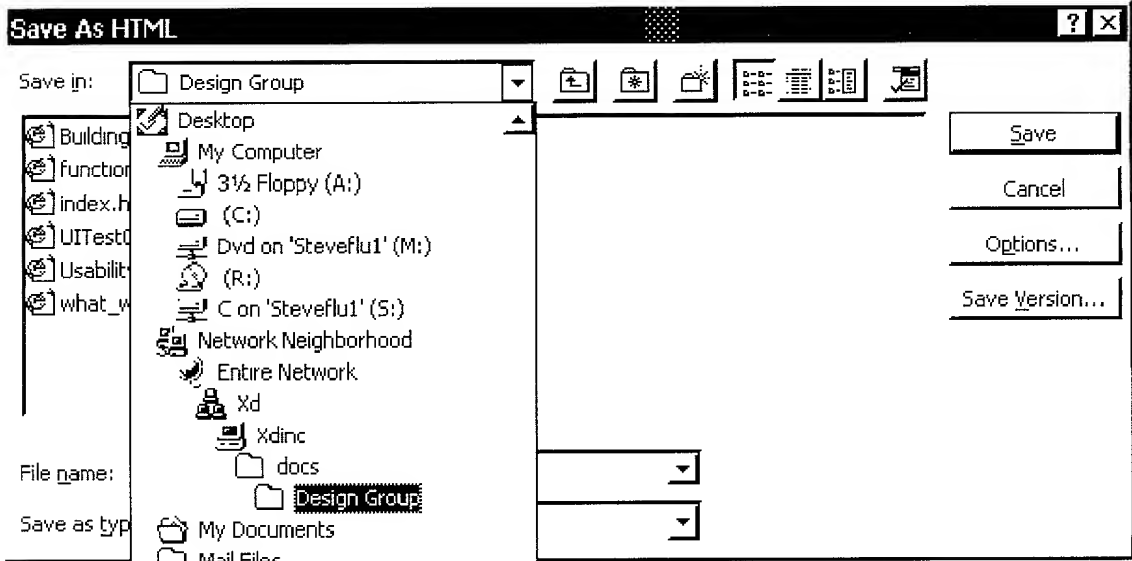
[001671] WIMP Examples:

The examples above illustrate different presentations of confirmations in the Windows environment. Note that in the example on the right, the confirmation Building Block has been combined with other Building Blocks to provide additional functionality.



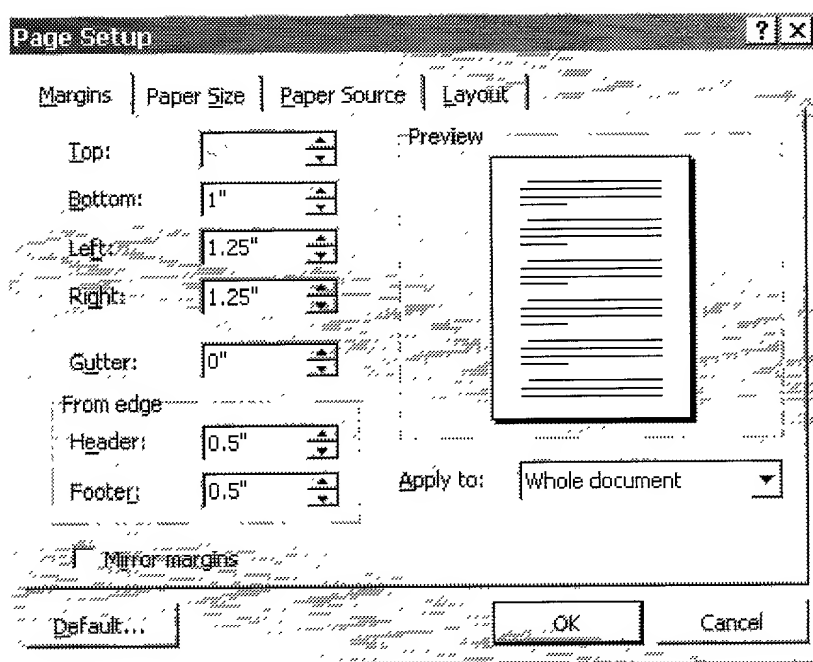


LISTS



Description	WIMP Examples	WIMP Deficits
A set of like ordered choices	List box, Combo box,	Pointing Scrolling Real estate

[001672] A spin control, which presents elements of an ordered set one by one, is one example of a list.



Associated Verbs	Description	Alternatives
Breadth Focus Manipulation	Moving the focus (on an element in the list) in a procedural way First/Last/Next/Prev/, mouse pointer,	
Exclusive Selection  Highlight/ Marking	Identifying a single element of the list, to the exclusion of all other elements	List is read to user (this is revealing hidden elements), listen for choice, indicate "yes/no" Speak choice AutoFill by character Grid control Apparently clairvoyant suggestions Keyword navigation

		Labels (e.g. alias "A", "B")
Inclusive selection	Identifying multiple elements of the list	This could be the same as the previous two until a certain keyword or action is initiated such as saying "Done."
Reorder	Changing the sort sequence of the list	
Create/ Delete	Modifies the set. Add a new element to the list / Remove an element from the list	
Copy		
Invoke selection(s) – default function	Where there is a single or primary function to perform on elements of the list; the act of triggering that function	
Perform Function on selection(s) – alternate associated function invocation	Where there are multiple functions to perform on elements of the list; the act of triggering a specific function	

#### COMMANDS

Description	WIMP Examples	Deficits w/WIMP
Using a command, the user initiates a new thread of execution. Icons,	Toolbar buttons, program icons	Fine motor control, screen real-estate

Description	WIMP Examples	Deficits w/WIMP
A Hierarchy is a collection of elements and lists that has two relationships. That of breadth, which lists have, and depth, which relates multiple lists.	Tree control menus	Lack of consistency

Description
The choice of any alpha-numeric, or special characters.

Description	WIMP Examples	Deficits w/WIMP
Notifications provide information to the user that is not associated with a choice.	Progress bar (no ack)	Cognitive load, screen real estate

Description	WIMP Examples	Deficits w/WIMP
Prompts surface choices to the user.	Any onscreen control that can be manipulated by the user. The text part of a dialog box. Earcon.	Reading requires continuous attention Audio is always

	Question Mark Icon	foreground
--	--------------------	------------

#### PC FEEDBACK

Description	WIMP Examples	Deficits w/WIMP
The PC presents indications on choices the user has made, or is currently making.	Moving the mouse	

#### HOW ARE BUILDING BLOCKS GROUPED?

##### FUNCTIONS

- [001673] The atomic functional elements of the User Interface, such as those defined in the previous section.

##### TASK

- [001674] A Task is a specific piece of work to be accomplished.
- [001675] In some embodiments, tasks are characterized with the following.
- Presence – This characterizes the quality of attention that the user should devote to the task. It may be Focus, routine, or awareness. See Divided User Attention.

Complexity – includes breadth and depth of orientation

Urgency/Safety – See ...

Exclusivity – The property of being difficult to do more than one of this kind of task. Example is phone conversations. See “Modality”

##### APPLICATIONS

- [001676] Tasks grouped by convenience.

## THREADS

- [001677] A Thread is a path of choices with a common user goal. The path can be tracked at a variety of levels, especially the Task or Application level.

## ENVIRONMENT

- [001678] The UI shell.

## WHAT ARE GENERAL UI ATTRIBUTES?

- [001679] General UI Attributes are abstractions belonging to, or characteristics of, a User Interface as a whole. Examples include the following:

### LEARNABILITY

#### EXPLORABILITY

##### FREEDOM

##### SAFETY

##### GROUNDING

##### CONSISTENCY

##### INVITATION (PARKS)

##### FAMILIARITY

##### MEMORABLE

##### PREDICTABILITY

#### SURFACING INFORMATION / CONTROLS

#### MENTAL MODELS

##### METAPHOR: SYMBOL SUGGESTING A REAL WORLD

##### OBJECT, IMPLYING MEANING.

##### SIMILE: DIFFERENT SYMBOLS TREATED AS HAVING LIKE

##### ATTRIBUTES OR INTERACTIONS.

#### DIRECT MANIPULATION

- [001680] By treating certain classes of visual elements as “objects” that have common interactions, used to surface common properties, (simile) we create a

mental model of being able to directly manipulate these “objects”, making interaction more learnable and memorable.

TRANSFERENCE

CONSISTENCY/PREDICTABILITY

CONSISTENCY W/ UNDERLYING ARCHITECTURE

[001681] Surface reality of underlying architecture.

MALLEABILITY: HOW ADAPTABLE A MENTAL MODEL IS TO  
BEING INTERPRETED AS A DIFFERENT BUT VIABLE  
MENTAL MODEL.

SINGLE MODEL OF COMMAND

[001682] Not a modal User Interface based on I/O modality.

NATURAL / INTUITIVE

SIMPLICITY

AVOIDANCE OF MODES

DIRECTNESS

AVOIDANCE OF ABSTRACTION

AVOIDANCE OF IMPLYING INFORMATION

AVOIDANCE OF SUPERFLUOUS INFORMATION

FLEXIBILITY

ADAPTABILITY

ACCOMMODATION

DEFERABILITY

[001683] Back burner / front burner – defer / activate

EXTENDABILITY  
EFFECTIVENESS  
EFFICIENCY  
EFFORT  
SAFETY  
ABILITY TO WITHDRAW FROM INTERACTION  
ERROR PREVENTION / RECOVERY  
FORGIVENESS  
HELP

## SYNCHRONIZING COMPUTER-GENERATED IMAGES WITH REAL WORLD IMAGES

[001684] In some situations, UIs are dynamically modified so as to display information in accordance with a real-world view without using real-world physical markers. In particular, the system displays virtual information on top of the user's view of the real world, and maintains that presentation while the user moves through the real world.

[001685] Some embodiments include a context-aware system that models the user, and uses this model to present virtual information on a display in a way that it corresponds to the user's view of the real world, and enhances that view.

[001686] In one embodiment, the system displays information to the user in visual layers. One example of this is a constellation layer that displays constellations in the sky, based on the portion of the real-world sky that the user is viewing. As the user's view of the night sky changes, the system shifts the displayed virtual constellation information with the visible stars. This embodiment is also able to calculate & display the constellation layer during the day, based on the user's location and view of the sky. This constellation information can be organized in a



virtual layer that provides the user ease of use controls, including the ability to activate or deactivate the display of constellation information as a layer of information.

[001687] In a further embodiment, the system groups various categories of computer-presented information related to the commonality of the information. In some embodiments, the user chooses the groups. These groups are presented to the user as visual layers. These layers of grouped information can be visually controlled (e.g., turned off, or visually enhanced, reduced) by controlling the transparency of the layer.

[001688] Another embodiment presents information about nearby objects to the user, synchronized with the real world surroundings. This information can be displayed in a variety of ways using this layering technique of mapping virtual information with the real-world view. One example involves enhancing the display of ATMs to a user searching for ATMs. Such a layer could be displayed in a layer showing streets and ATM locations, or such a layer could display the ATM's location near the user. Once the user has found the ATM being desired, the system could turn off the layer automatically, or based on the user's configuration of the behavior, simply allow the user to turn off the layer.

[001689] Another embodiment displays a layer of information, on top of the real-world view, that shows information represents the path the user traveled between different points of interest. Possible visual clues (bread crumbs) could be any kind of visual image, like a dashed line, or dots, to represent the route, or path, the user traveled. One example involves a user searching a parking garage for a lost car. If the user cannot remember where the car is parked, and the user is searching the parking garage, the system can trace the search-route and help the user avoid searching the same locations by displaying that route. In a related situation, if the bread-crumbs trail was activated when the user parked the car, the

user could turn on that layer of information and follow the virtual trail as it displays to the user in real-time, adjusting to the user's view, thus leading the user directly back to the parked vehicle. This information could also be displayed as a bird's-eye view, showing the path of the user relative to a map of the garage.

[001690] Another embodiment displays route information as a bird's-eye view showing a path relative to a map. This information is presented in overlaid, transparent, layers of information and can include streets, hotels and other similar information related to a trip.

[001691] The labeling and selection of a particular layer can be provided to the user in a variety of methods. One example provides labeled tabs, like on hanging folders that can be selected by the user.

[001692] The system accomplishes the task of presenting virtual information on top of real-world information by various means. Three main embodiments are tracking head positions, tracking eye positions, and real world pattern recognition. The system can also use a combination of these aspects to obtain enough information.

[001693] The head positions can be tracked by a variety of means. Three of these are inertial sensors mounted on the user's head, strain gauges, and environmental tracking of the person. Inertial sensors worn by the user can provide information to the system and help it determine the real-world view of the user. An example of inertial sensors is some kind of jewelry to detect the turns of a user's head. Strain gauges, for example, embedded in a hat, or the neck of clothing, measure two axes: left and right, along with up and down. The environment can also provide information to the system regarding the user's head and focus. The environment can provide pattern-matching information of the user's head to help indicate the visual interest of the user. This can occur from a camera watching head movements, like in a kiosk or other such booth, or any camera that can provide information about the user. Environmental sensors can

perform triangulation based on a single beacon, or multiple beacons, transmitting information about the user, and the user's head & eyes. The sensors of a room, or say a car, can triangulate information about the user and present that information to the system for further use by the system for determining the user's view of the real-world. Also, the reverse works, where the environment broadcasts information about location, or distance from one the sensors in the environment, such that the system can perform the calculations without needing to broadcast information about the user.

[001694] The user's system can track the eye positions of the user for use in determining the user's view of the real world, which can be used by the system to integrate the presentation of virtual information with the user's view of the real world.

[001695] Another embodiment involves the system performing pattern recognition of the real world. The system's software dynamically detects the user's view of the real world and incorporates that information when the system determines where to display the virtual objects such that they remain integrated while the user moves about the real world.

[001696] Those skilled in the art will also appreciate that in some embodiments the functionality provided by the routines discussed above may be provided in alternative ways, such as being split among more routines or consolidated into less routines. Similarly, in some embodiments illustrated routines may provide more or less functionality than is described, such as when other illustrated routines instead lack or include such functionality respectively, or when the amount of functionality that is provided is altered. In addition, those skilled in the art will appreciate that the data structures discussed above may be structured in different manners, such as by having a single data structure split into multiple data structures or by having multiple data structures consolidated into a single

data structure. Similarly, in some embodiments illustrated data structures may store more or less information than is described, such as when other illustrated data structures instead lack or include such information respectively, or when the amount or types of information that is stored is altered.

[001697] From the foregoing it will be appreciated that, although specific embodiments have been described herein for purposes of illustration, various modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims and the elements recited therein. In addition, while certain aspects of the invention are presented below in certain claim forms, the inventors contemplate the various aspects of the invention in any available claim form. For example, while only some aspects of the invention may currently be recited as being embodied in a computer-readable medium, other aspects may likewise be so embodied. Accordingly, the inventors reserve the right to add additional claims after filing the application to pursue such additional claim forms for other aspects of the invention.